NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

**NETWORK SECURITY AND
THE NPS INTERNET FIREWALL**

by

Jody L. Schivley

September 1994

Thesis Advisor: Timothy Shimeall

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution is unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: Computer Science Dept. Naval Postgraduate School; 6b. OFFICE SYMBOL (*if applicable*): CS	7a. NAME OF MONITORING ORGANIZATION: Naval Postgraduate School
6c. ADDRESS (*City, State, and ZIP Code*): Monterey, CA 93943-5000	7b. ADDRESS (*City, State, and ZIP Code*): Monterey, CA 93943-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION; 8b. OFFICE SYMBOL (*if applicable*)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (*City, State, and ZIP Code*)	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (*Include Security Classification*)
Network Security and the NPS Internet Firewall (Unclassified)

12. PERSONAL AUTHOR(S)
Schivley, Jody L.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (*Year, Month, Day*)	15. PAGE COUNT
Master's Thesis	FROM 09/92 TO 09/94	1994, September, 16	120

16. SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the United States Government.

17. COSATI CODES			18. SUBJECT TERMS (*Continue on reverse if necessary and identify by block number*)
FIELD	GROUP	SUB-GROUP	Network, Security, Firewall, TCP/IP, Packets, NFS, NIS, DDN, Internet

19. ABSTRACT (*Continue on reverse if necessary and identify by block number*)

As the Naval Postgraduate School's (NPS) computer network continues to incorporate computers with a wide variety of security holes, it is vital that an Internet firewall be installed to provide perimeter security for NPS from the Internet. NPS has had systems compromised by unauthorized individuals who have gained access via the Internet.

The approach taken by this thesis was to analyze the type of Internet firewalls available and chose a design that provides the protection required at NPS while maintaining the Internet functionality desired. After choosing the appropriate type of firewall, it was tested for functionality and performance. The functionality test successfully validated that the bootp, netwall, tftp, sunrpc, and nfsd packets could be blocked while other network services remained functional. The performance testing process first monitored existing traffic to and from the BARRNET and DDN routers. The second step determined the firewall's performance with a well known network measurement tool, New Test TCP/IP (*nttcp*). The existing data rates to and from the Internet are on average 438 kilobits per second and the *nttcp* tests showed that the firewall could run at 600 kilobits per second. These results validated that the firewall could maintain the data rates currently required to the Internet.

This thesis resulted in a firewall, obtained from Texas A&M, that can be installed and used to improve the network security between the NPS network and the Internet. This firewall runs on a PC and would be located between the NPS network and the BARRNET and DDN routers. This would result in a perimeter of security for the NPS network, to assist in the ever growing need for network security.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION	
[X] UNCLASSIFIED/UNLIMITED [ ] SAME AS RPT. [ ] DTIC USERS	UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL: Prof Timothy Shimeall	22b. TELEPHONE (*Include Area Code*): (408) 656-2509	22c. OFFICE SYMBOL: CS/Sm

Approved for public release; distribution is unlimited

NETWORK SECURITY AND THE NPS INTERNET FIREWALL

Jody L.. Schivley
BS Computer Science, Northern Arizona University, 1984

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 1994

Author: Jody L. Schivley

Approved By: Timothy Shimeall, Thesis Advisor

G. M. Lundy, Second Reader

Ted Lewis, Chairman,
Department of Computer Science

ABSTRACT

As the Naval Postgraduate School's (NPS) computer network continues to incorporate computers with a wide variety of security holes, it is vital that an Internet firewall be installed to provide perimeter security for NPS from the Internet. NPS has had systems compromised by unauthorized individuals who have gained access via the Internet.

The approach taken by this thesis was to analyze the type of Internet firewalls available and chose a design that provides the protection required at NPS while maintaining the Internet functionality desired. After choosing the appropriate type of firewall, it was tested for functionality and performance. The functionality test successfully validated that the bootp, netwall, tftp, sunrpc, and nfsd packets could be blocked while other network services remained functional. The performance testing process first monitored existing traffic to and from the BARRNET and DDN routers. The second step determined the firewall's performance with a well known network measurement tool, New Test TCP/IP (*nttcp*). The existing data rates to and from the Internet are on average 438 kilobits per second and the *nttcp* tests showed that the firewall could run at 600 kilobits per second. These results validated that the firewall could maintain the data rates currently required to the Internet.

This thesis resulted in a firewall, obtained from Texas A&M, that can be installed and used to improve the network security between the NPS network and the Internet. This firewall runs on a PC and would be located between the NPS network and the BARRNET and DDN routers. This would result in a perimeter of security for the NPS network, to assist in the ever growing need for network security.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

A. INTRODUCTION

Network security is the means of defending a domain (a collection of interconnected computers) against unwanted disclosure, modification, or usage. Network security plays a vital role in computer security. Individual computers on a network are susceptible to many threats. Effective network security provides a level of security across the board to all the computers connected to that network, reducing the total number of threats imposed on each computer.

Network security is a relevant topic in many organizations. The widespread concern over network security is due to the connectivity of many organizations' internal computers to one another as well as the connection of the organization's network as a whole to the Internet.

For over 20 years the Internet has become a very important part of communications [STAN 93]. The Internet is a large network of networks that includes networks in 100 countries, with over one million computers and 10 million users. It is so large and rapidly growing, the actual number of computers and users can not be accurately counted. With the enormous growth of the Internet, networks connected to it become as vulnerable as they are valuable.

There are people who intentionally try to break into systems connected to the Internet. There have been many security problems and break-ins on the Internet, such as the KGB Hackers, the Morris Worm, and the Nova Penetration [RUSS 91]. The attacks can be automated or manual. Automated attacks, once started, proliferate on their own. They spread rapidly and may infiltrate many systems before they are eradicated. Manual attacks are generally system break-ins with the intent of obtaining specific information. The manual attacks can be hard to detect and can be overlooked or go undetected. Real network security comes from being alert and recognizing the real and potential threats to the network as well as how to monitor and combat them [CARL 92].

The Naval Postgraduate School (NPS) must take a more active role in network security due to its ever increasing connectivity.

B. BACKGROUND

In late 1969, the first large-scale data network was originated. This network was sponsored by the Advanced Research Projects Agency (ARPA) and appropriately named the Advanced Research Projects Agency Network (ARPANET). ARPANET connected geographically-separate military, university, and research computer systems and revolutionized computer communications [RUSS 91].

The Internet, as we know it today, began to take shape in the 1970's. ARPA converted machines to use a protocol suite that allows systems on different networks to communicate. This protocol suite consists of two major parts, the Transmission Control Protocol and the Internet Protocol (referred to as TCP/IP). The TCP/IP protocol suite became entrenched into the Internet and is still used today.

In 1984, the ARPANET split into two large networks. One part of the ARPANET connected primarily university and research computers and has become the backbone of the Internet that exists today. The other branch of the ARPANET is a collection of military networks known as the Defense Data Network (DDN).

Interest in network security has grown significantly in the computer community. The Internet itself provides no security. Only a small number of companies provide network security services or products. At this point in time, it is up to each individual network connected to the Internet to provide its own line of defense to protect itself from unauthorized access.

C. SECURITY PROBLEMS IN NETWORKS

The advantages gained from using a network of computers also bring disadvantages. Networks have security problems because of the following reasons [PFLE 89]:

- *Sharing* - Both data and resources are shared among users on a network.

- *Complexity of system* - There are far more components involved with a network versus a stand-alone system.
- *Unknown perimeter* - With connectivity to the Internet the bounds of the network do not end within the organization. They span the world.
- *Many points of attack* - The unknown perimeter leads to many points of attack.
- *Unknown path* - On a network of systems, a user can gain access to a particular system through hundreds or thousands of different routes.

This leads to a real need for network security. The primary goals of computer security are to ensure secrecy, integrity, and availability [PFLE 89]. These same goals must also be the primary goals in achieving network security.

Network security is extremely complex because of the number of threats that must be addressed. A threat may be a person, thing, or event that may compromise the secrecy, integrity, or availability of network assets. The threats may not be intentional or they may be deliberate. Unintentional threats include human error, equipment failure, natural disasters, or communication failures. Deliberate threats include theft, vandalism, sabotage, and misuse of resources [STAN 93].

The threats are compounded by the number of individuals and computer systems using the network. The computer network is seen as an extension of the individual's computer. Each individual has a different level of education on computer system security and network security. Every user also has a different perception of the proper use of network assets. Some users feel that any resource they can obtain is theirs to use, whether they are authorized or not. On the other hand, the administrators must make sure the network assets are always available to authorized users.

Network security should constitute an overall plan and policy for the security of the systems on the network. Ideally, the network security plan and policy should be formulated prior to network implementation and periodically reviewed and updated to reflect changing conditions. Unfortunately, most organization's network security plans and policies lag implementation.

The problems dealing with network security span all aspects of the computing community. The corporate world, government agencies, and educational institutes are all fighting this uphill battle.

There are many defensive measures that can be taken to increase network and system security. Some measures are listed below [STAN 93]:

- *Perform backups* - Good backups can save time and money. They must be done efficiently and stored properly.
- *Train users* - User awareness of good computer security practices can help users help themselves.
- *Set policies* - The network and computer security policies are important. They inform the administrators and users of how the organization views security.
- *Physical access controls* - Limit the physical access to rooms where the servers are kept. Also, lock down public workstations to the table tops.
- *Logical access controls* - Enforce the use of identification and authentication, then restrict users to authorized activities and resources.
- *Operational controls* - Make sure that operational procedures are well documented and evaluated on a regular basis.
- *Log, monitor, and report* - It is important to audit and evaluate who is performing critical events and when they are being performed.

Unfortunately, experience has shown that administration of all the measures listed above on every computer is costly or requires too many resources. A solution to this problem is to augment these security measures with security measures under a central point of control. An effective measure, that can be centrally managed, is an Internet firewall. A firewall is a collection of components used to protect one network from another untrusted network, such as the NPS network and the Internet. This measure can limit the damage caused by eavesdropping or by flooding attacks from the Internet. It can also provides logging and monitoring of events. Firewalls help isolate physical network failure as well, keeping most segments operational [STAN 93].

D. NAVAL POSTGRADUATE SCHOOL'S NETWORK SECURITY POLICY

NPS is an academic institution whose emphasis is on advanced professional studies and research programs that are related to the Navy's interest, as well as the interests of other

arms of the Department of Defense (DoD). Military officers and defense officials from all services and other nations attend the school.

To fulfill its mission, NPS strives to be responsive to technological change and to prepare officers to introduce and utilize future technologies. To meet this mission, NPS is on the leading edge of network and computer technology. The networks and computers at NPS provide an infrastructure used by the students, faculty, and staff for learning and advanced research.

The diverse user population at NPS creates a challenge for administrators to enforce network and computer security. The resources are used by individuals with extensive experience and others with no experience at all. A lack of education in network and computer security has caused problems. There is also a delicate balance between the users' functional requirements and properly implemented security measures. The security measures, by definition, are put in place to protect the computer and network assets but users frequently feel they are too restrictive.

Budget cuts are a reality that must be addressed. All government agencies have less money to spend on resources, including salaries to hire more administrators.

Computer and network security at NPS is regulated by the DoD and the Department of the Navy (DoN). The school also has an internal network security policy. This policy is in Appendix A and covers the following areas:

- *User Identification* - A central approach to assigning every authorized user a unique user name.
- *System Access* - Restricted use of the root privileges and guest, class, or visitor accounts on systems attached to the network
- *File Permissions/Data Security* - Instruction on initial user directory permissions and a provision that system administrators must provide an encryption tool for users.
- *Data Integrity* - Describes backup procedures.
- *Security Monitoring and Reporting* - Defines the level of monitoring and reporting to be done by system administrators.
- *Classified Information* - States that no classified information will be stored or processed on any NPS system on the network.
- *Safeguarding Network Operability* - Restricts connections to the NPS backbone to only those systems approved by the Computer Center.

- *Responsibilities for ADP Security* - Put the responsibility of computer security into the performance evaluation critical elements, to help enforce or reward compliance.

The NPS Network Security Policy is yet to be fully implemented. An unimplemented policy provides no defense in network security. Additionally, network security is far too complex for a policy alone. The more exposed a network is, the more vulnerable it is to threats. Therefore, the network security posture at NPS must be robust.

E. CURRENT NETWORK SECURITY POSTURE

Currently, the computer network at NPS is connected to the Internet with no central funnel for inbound data to be passed through and as a result, all NPS computers are vulnerable to all outside threats. These threats require each system on the network to maintain a high level of security, which is a monumental task. There are hundreds of computers at NPS, with many system administrators, who possess varying levels of experience. The operating systems across campus vary significantly and contain numerous bugs and security holes, making it almost impossible for a consistent level of security across the NPS network.

Network security should not be a reactionary process, rather it should be a methodical process. The network security policy currently in place at NPS should be supplemented with some centrally managed and enforced network security policy.

F. PROBLEM STATEMENT

The Internet provides information that would be difficult or impossible to acquire by other means. There is no dispute that the connectivity NPS has to the Internet is vital to the mission of the school. This thesis will explore how the school can continue to benefit from the Internet connection while protecting the network assets from the threats the Internet connection creates.

The issues and questions that will be covered by this thesis work are:

- What is the operational definition of network security in an environment like NPS? NPS is in a unique position, since it is an educational institute but it also a government

agency that is required to follow DoD and DoN guidelines. An operational definition of network security will establish how NPS fits into both roles.

- What network security model best suits NPS? This model would define what threats must be eliminated to provide the proper level of network security while serving the varied needs of the schools academic staff and mission.
- Would a network firewall be effective in eliminating or reducing the current threats while sustaining the users functionality? The firewall, a popular solution, is an option that will be thoroughly investigated.
- If a network firewall would be effective, what would be the most efficient and most cost effective implementation? The effective cost of a solution must be researched and compared to the effective cost of other possible solutions.

These are important questions from a scientific point of view. Every environment is different and the network security policy should reflect these differences. Just as there are different policies, there are different ways to implement these network security policies. It is important to explore each possibility and find the best solution for NPS.

G. OVERVIEW OF THE THESIS

The main thrust of this thesis is to properly define and model the network security policy at NPS and then discuss the appropriate perimeter security to protect the NPS network from the Internet. First, a functional network security policy will be established based on DoD and DoN requirements.

Second, a network security model will be developed. This will be accomplished by determining the minimum requirements necessary at NPS. Since NPS is a government organization, there are DoD Directives that must be followed. The risk assessment procedure is used to determine the minimum evaluation class required for automated information systems, including networks, based on the sensitivity of the information present and on the clearance of its users.

Third, once the security network policy and model are established, this thesis will discuss Internet firewalls which are tools that could, potentially, provide the required network security. A general discussion of firewalls will be included, followed by functionality and performance testing of the selected firewall architecture in a proof of

principle environment. The test procedures used will be outlined and the test results discussed.

II. SECURITY / THREAT MODEL

A. FUNCTIONAL NETWORK SECURITY DEFINITION

The formal definition of network security according to the National Computer Security Center (NCSC) is as follows:

> Network security is the protection of networks and their services from unauthorized modification, destruction, or disclosure. Providing an assurance that the network performs its critical functions correctly and there are no harmful side-effects. Includes providing for information accuracy [NCSC 87].

Within the Department of Defense, the security requirements of computing systems is outlined in the *Department of Defense Trusted Computer System Evaluation Criteria* (TCSEC), informally known as the "orange book" [DODD 85]. The rating structure in the TCSEC has been extended to networks of computers. It consists of four basic divisions, A, B, C, and D. Division A stipulates the most comprehensive degree of security down to division D which specifies no security requirements [MADR 92].

Molding these formal definitions and outlines around the mission of the Naval Postgraduate School is a difficult challenge. NPS is an academic institute whose emphasis is on advanced professional studies and research programs. The school is responsive to technological change and is on the leading edge of network and computer technology. This environment is not conducive to structured network security. There are researchers who do not want to be concerned with network security and at the other end of the spectrum military officers who are required to implement the network security.

Network security is as much a technical issue as it is a management issue. Just below the highest management at NPS, most of the computing assets at NPS are divided between two large branches, each under different middle management. Naturally, the organizational dynamics in these two groups is different and the management of these groups has shown no active concern in computer or network security. This is an issue confronting many organizations today.

So, with this in mind, how does the school functionally define its network security policy and is this policy different from the policy established by NCSC? The actual network security policy at NPS is no different from that established by NCSC; however, the implementation is where the difference lies. The first step is to determine the minimum requirements for a network security policy at NPS. Since NPS is a government organization, there are DoD Directives that must be followed. A formal risk assessment procedure is used to determine the minimum evaluation class required for automated information systems, including networks, based on the sensitivity of the information present and on the clearance of its users.

B. RISK MANAGEMENT

Risk management is a methodology used to identify, measure, and control events which adversely affect security. It involves cost-benefit analyses to ensure appropriate cost-effectiveness of security safeguards. A risk management program is mandated by Enclosure (3) of DoD Directive (DODD) 5200.28 [NCSC 87].

Risk assessment is the procedure used to estimate potential loss that may result from system vulnerabilities and to quantify the damage that may result if certain threats occur. DODD 5200.28 Enclosure (4) mandates the use of a methodology, extracted from the TCSEC Environments Guideline, to determine the recommended evaluation class based on a specific environment. The Directive also recommends this method to determine minimum computer-based requirements in a network. Use of a different method requires prior approval of the Assistant Secretary of Defense (ASD) Command, Control, Communications, Computers and Intelligence (C^4I).

DODD 5200.28 enclosure (4) contains six major steps in the risk assessment procedure. These steps are listed below and were the steps used to determine the minimum evaluation class for the NPS network.

- *Step 1* - Determine system security mode of operation.
- *Step 2* - Determine minimum user clearance or authorization rating.
- *Step 3* - Determine maximum data sensitivity rating.

- *Step 4* - Determine risk index.
- *Step 5* - Determine minimum security evaluation class for computer-based controls.
- *Step 6* - Determine adjustments to computer security evaluation class required.

1. Step 1 - Determine System Security Mode of Operation

A security mode is a mode of operation in which an organization has been accredited to operate. These security modes each contain restrictions on the user clearance levels, formal access requirements, need-to-know requirements, and the range of sensitive information permitted on the equipment [DODD 88].

a. Dedicated Security Mode

A mode of operation where all users have the clearance or authorization and need-to-know for all data handled.

b. System High Security Mode

A mode of operation where all users having access possess a security clearance or authorization, but not necessarily a need-to-know.

c. Multilevel Security Mode

A mode of operation that allows two or more classification levels of information to be processed simultaneously within the same system when not all users have a clearance or formal access approval for all data handled.

d. Partitioned Security Mode

A mode of operation where all personnel have the clearance, but not necessarily formal access approval and need-to-know, for all information handled.

The computers and network at NPS have not been accredited to operate at any of the modes of operation listed above. The systems on campus that are connected to the network handle sensitive unclassified information and unclassified information. Neither type of information fits into the modes described above, however it must be safeguarded. Sensitive unclassified information is any information the loss, misuse, or unauthorized access to, or modification of which, might adversely affect U.S. national interest, the conduct of DoD programs, or the privacy of DoD personnel. Unclassified information does

not need to be safeguarded against disclosure, but must be safeguarded against tampering, destruction, or loss due to record value, utility, replacement cost or susceptibility to fraud, waste, or abuse.

2. Step 2 - Determine Minimum User Clearance or Authorization Rating

The minimum user clearance or authorization rating (R_{min}) is defined as the maximum clearance or authorization of the least cleared or authorized user. To determine a minimum clearance or authorization rating Table 1 was used [DODD 88].

Minimum User Clearance	Rating (R_{min})
Uncleared OR Not Authorized (U)	0
Not Cleared but Authorized Access to Sensitive Unclassified Information (N)	1
Confidential (C)	2
Secret (S)	3
Top Secret (TS) and/or current Background Investigation (BI)	4
TS and/or current Special Background Investigation (SBI)	5
One Category (1C)	6
Multiple Categories (MC)	7

Table 1: Rating Scale For Minimum User Clearance (R_{min})

The value assigned to R_{min} would be 1 for the NPS network.

3. Step 3 - Determine Maximum Data Sensitivity Rating

The maximum data sensitivity rating (R_{max}) is defined as the rating of the most sensitive data. To determine a maximum data sensitivity rating Table 2 was used [DODD 88].

The value assigned to R_{max} would be 2 for the NPS network. There are personal computers and workstations on the network which handle data concerning grades, salaries, promotions, and performance appraisals, which is sensitive unclassified information.

Maximum Sensitivity Rating without Categories	Rating (R_{max})	Maximum Data Sensitivity with Categories	Rating (R_{max})
Unclassified (U)	0	N/A	
Not Classified but Sensitive (N)	1	N with one or more Categories	2
Confidential (C)	2	C with one or more Categories	3
Secret (S)	3	S with one or more Categories only one Category containing S S with two or more categories containing S	4 5
Top Secret (TS)	4	TS with one or more Categories only one Category containing S or TS TS with two or more Categories containing S or TS	6 7

Table 2: Rating Scale For Maximum Data Sensitivity (R_{max})

4. Step 4 - Determine Risk Index

The disparity between the minimum clearance or authorization (R_{min}) and the maximum data sensitivity (R_{max}) is what determines the risk index. The risk index is computed as follows [DODD 88].

a. Case a

If the value of R_{min} is less than the value of R_{max}, then:

Risk Index = R_{max} - R_{min}

b. Case b

If R_{min} is greater than or equal to R_{max}, then:

If there are categories to which some users are not authorized access, Risk Index equals 1. Note that a category is a grouping of classified or sensitive unclassified information to which an additional restrictive label is applied for signifying that personnel are granted access to the information only if they have formal access approval or other applicable authorization.

In all other cases, Risk Index = 0.

Since it has been determined that R_{min} =1 and R_{max}= 2, the value assigned to the Risk Index is 1 for the NPS network.

5. Step 5 - Determine Minimum Security Evaluation Class

Table 3 shall be used to determine the minimum security class required based on the computed risk index, above [DODD 88].

Risk Index	Security Mode	Minimum Security Class
0	Dedicated	No Minimum Class
0	System High	C2
1	Multilevel, Partitioned	B1
2	Multilevel, Partitioned	B2
3	Multilevel	B3
4	Multilevel	A1
5	Multilevel	*
6	Multilevel	*
7	Multilevel	*

Table 3: Minimum Security Class Based On Risk Index

Step 4 determined that the Risk Index = 1 for the NPS network. Using that value and Table 3 reveals that the minimum security class required for the NPS network is B1 level security.

6. Step 6 - Adjustments to Computed Security Evaluation Class Required

There are additional requirements or recommendations relevant to determining the minimum evaluation class. The requirements include the following [DODD 88]:

- Where a network is used, care should be taken to ensure that the requirements for accreditation of the automated information systems (AIS) are not violated due to the presence of the network technology.
- In the dedicated mode where the AIS is connected to a network or to another AIS, is it recommended at least level C1 be used.
- An AIS processing in one or more security modes and/or at one or more security levels for certain periods of time, there may be a need for more than one risk index.

Taking into consideration the possible adjustments, none apply to the NPS network.

C. CHARACTERISTICS OF B1 LEVEL SECURITY

The risk assessment in the previous section determined the NPS network should be running at the B1 level of security. The security class structure outlined in the TCSEC was originally designed for stand-alone AIS and more recently was extended to networks of computers. It consists of four basic classes, A, B, C, and D and some of these classes have subclasses. Class A1 stipulates the most comprehensive degree of security down to class D which specifies no security requirements.

Class C1 provides discretionary security protection. Network systems in this class nominally satisfy the discretionary security requirements by providing separation of users and data. Class C2 provides controlled access protection. This class enforces a finer granularity of discretionary access control than C1 networked systems and makes users individually accountable for their actions through login procedures, auditing of security-relevant events, and resource isolation. Class B1 network systems provides labeled security protection and requires all the features required for class C2. In addition, an informal statement of the security policy model, data labeling, and mandatory access control over subjects and storage objects must be present [NCSC 87]. There are three major areas of class B1 networked systems:

1. Security Policy

An overall network security polity must be enforced for B1 level security. The policy must be an access control policy having two primary components: mandatory and discretionary. The mandatory policy must define the set of distinct sensitivity levels that it supports. This policy shall be based on the labels associated with the information that reflects its sensitivity with respect to secrecy and/or integrity, where applicable, and labels associated with users to reflect their authorization to access such information.

2. Accountability

The C2 class of network security shall maintain authentication data that includes information for verifying the identity of individual users as well as information for determining the clearance and authorization of individual users.

3. Documentation

For C2 level security, single summary, chapter, or manual must describe user visible protection mechanisms at the global (network system) level and at the user interface of each component, and the interaction among these.

The B1 rating is significantly less stringent than the B2 rating. The distinguishing difference is that the operating system developer might be able to add security measures to an existing operating system so it qualifies for a B1 rating. Security must be included in the design of the operating system for a B2 rating.

D. B1 LEVEL SECURITY AND THE NPS NETWORK

The network at the Naval School is a heterogeneous environment with computers from most of the popular vendors in the marketplace. There are multiuser computers and single user computers running a wide variety of operating systems. Table 4 is a partial listing of the computers connected to the NPS network.

Vendor	Computer Type	Operating System	Single User or Multiuser
Amdahl	Mainframe	VM / MVS	Multiuser
Cray	Super Computer	UNICOS (Unix)	Multiuser
Digital Equipment Corporation	Workstation	Unix	Multiuser
Digital Equipment Corporation	Super Mini	VMS	Multiuser
Hewlett Packard	Workstation	HPUX (Unix)	Multiuser
IBM	Workstation	Ultrix (Unix)	Multiuser
NEXT	Workstation	Unix	Multiuser

Table 4: Computers On The NPS Network

Vendor	Computer Type	Operating System	Single User or Multiuser
Macintosh	Personal Computer	Apple	Single User
Solbourne	Workstation	Unix	Multiuser
Sun	Workstation	SunOS / Solaris (Unix)	Multiuser
Silicon Graphics Incorporated	Workstation	IRIX (Unix)	Multiuser
Miscellaneous	Personal Computer	DR-DOS	Single User
Miscellaneous	Personal Computer	LINUX	Multiuser
Miscellaneous	Personal Computer	MS-DOS	Single User
Miscellaneous	Personal Computer	OS2	Single User
Miscellaneous	Personal Computer	Windows	Single User

Table 4: Computers On The NPS Network

It is difficult to obtain and maintain a comprehensive listing of all computers connected to the campus network. The mission at the school requires state of the art equipment so new equipment is arriving almost weekly from different vendors. There is no central automation authority to track what equipment is coming in and where it will be located. Each department is responsible for their own automation inventory.

With the diversity of vendors and operating systems associated with the NPS network the subject of security becomes a monumental task. Every vendor listed in Table 4 is not necessarily developing secure operating systems. Those vendors who are creating a secure operating environment are in different stages of development. This is in addition to the distinct difference in attitude towards automation and network security. As a government organization, NPS is required to provide security, specifically B1 level security as determined earlier. As researchers, the staff and faculty are focusing on their area of expertise. They use the automation as a tool in their research and most often do not have the time to dedicate to computer security.

It is important to move towards B1 level security across all systems on the NPS network. So, what options are available?

One option is to disconnect any computer on the NPS network that is not B1 compliant. This would immediately bring the network in compliance with the B1 level of security. However, this option is not feasible because there are no known systems connected to the NPS network running B1 level security, so all computers would be disconnected from the network.

Another option is to disconnect the systems on the network processing sensitive unclassified information and only allow unclassified nonsensitive information processing on the systems connected to the network. This would include removing the mainframe from the network and any other systems processing information such as grades, salaries, or performance appraisals. This option is not feasible because it places an unacceptable burden on the departments.

A third option is to ignore network security all together. This is infeasible because intrusions into the NPS network have been discovered coming from locations outside the school. These intrusions have been responsible for denial of service to NPS users, invasion of privacy, possible data corruption, and use of NPS computers as a base of attack.

An industry-wide approach that promotes network security is to accept the risk associated with a lower level of security on the individual systems connected to the network while providing a higher level of security around the perimeter of this network. A basic level of trust would be given to the users inside the network at the school and anyone outside of the network would be considered a possible threat. This boundary could be made between connections to outside networks and the school's network. It would be a central location that could provide a better understanding of intruders, it could identify specific or potential threats, increase the ability to react to threats, and allow a central basis of control. The perimeter security is an approach that can be implemented promptly with minimal NPS staff or faculty.

III. NPS FIREWALL

A. POSSIBLE SAFEGUARDS FOR NETWORK SECURITY

At the end of Chapter II it was concluded that the best short-term approach to network security for NPS is perimeter security. This would be a boundary set up between the outside network and the internal NPS network. The outside is defined as the Internet and telephone dial-in access. At this time, the best perimeter security is a network firewall.

A firewall is a collection of components used to protect one network from another untrusted network, such as the NPS network and the Internet. Firewalls have the following properties:

1. Centralized Network Access

All traffic from inside to outside, and vice-versa, must pass through the firewall. Security on the NPS network is dramatically enhanced because there is one central location to manage network access. It is a good solution at large sites, such as NPS, where systems are physically spread out across a large area with many different system and network administrators are involved.

2. Allow and Deny Access

Only authorized traffic, as defined by the security policy, will be allowed to pass. A firewall would guard NPS at the front door only allowing approved packets onto the NPS network. The firewall can be set up to allow or impede network traffic in a manner that will best suit the school. It is extremely flexible and can dynamically change as computing and network needs change.

3. Firewall is More Secure

The firewall itself is immune to penetration because it will not accept commands from a network. Also, the administration of the firewall is done in a more security conscious manner than other hosts on the network.

Internet firewalls are a relatively new approach to network security. There are general guidelines and basic building blocks to be considered for every firewall but there are no cookbook instructions to follow.

B. COSTS OF IMPLEMENTING SECURITY CONTROLS

Any safeguard against network security is not free. In the case of a network firewall the costs include the following: [CHES 94]

- hardware purchase
- hardware maintenance
- software development or purchase
- administrative setup and training
- ongoing administration and trouble-shooting
- lost business or inconvenience from a broken gateway or blocked services
- the loss of some services or convenience that an open connection would supply

The costs above must be weighed against the costs of not having a firewall. These costs include [CHES 94]:

- the effort spent in dealing with break-ins, including the loss of business
- legal and other costs of sponsoring hacker activity

C. FIREWALL DESIGN DECISIONS

There are two basic firewall designs that need to be considered. The first is a proxy level gateway and the second is a packet screening filter. Before discussing these options it is necessary to understand the Transmission Control Protocol/Internet Protocol (TCP/IP) network protocol architecture, which is the predominate network protocol being used at NPS and the protocol used on the Internet today.

1. TCP/IP Protocol Architecture

In recent years, the Open Systems Interconnect (OSI) Reference Model was developed by the International Standards Organization (ISO) to describe the structure and function of data communications protocols. However, the OSI protocol suite is not commonly used

because the TCP/IP protocol architecture has been used for over 30 years and is embedded in most operating systems.

The TCP/IP protocol architecture contains four layers, represented like a pile of building blocks stacked upon one another. The structure is often called a stack or protocol stack. Figure 1 identifies each layer by name and contains a brief description of the function of that layer.

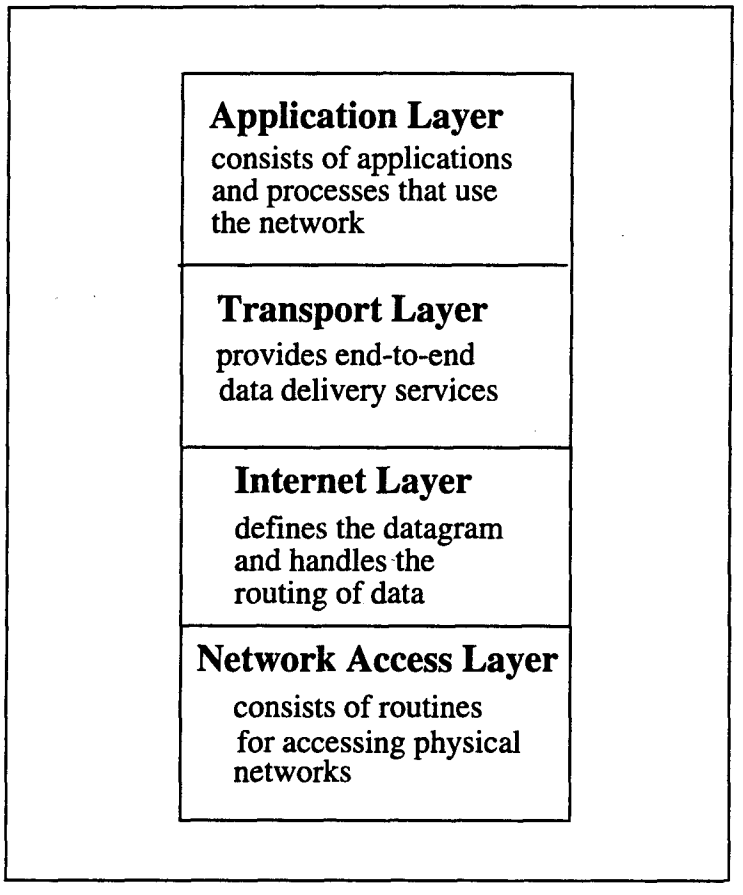

Figure 1: TCP/IP Protocol Architecture

Each layer does not define one protocol, it defines a function that may be performed by any number of protocols. Figure 2 illustrates the different protocols within each TCP/IP layer. Every protocol communicates with its peer. A peer is an implementation of the same protocol in the equivalent layer on a remote system. However, there is no direct

communication between peer layers. The layers send the data down to the next lower layer to get the data across to the peer layer. Each layer is unaware of the functionality of the layers above and below.

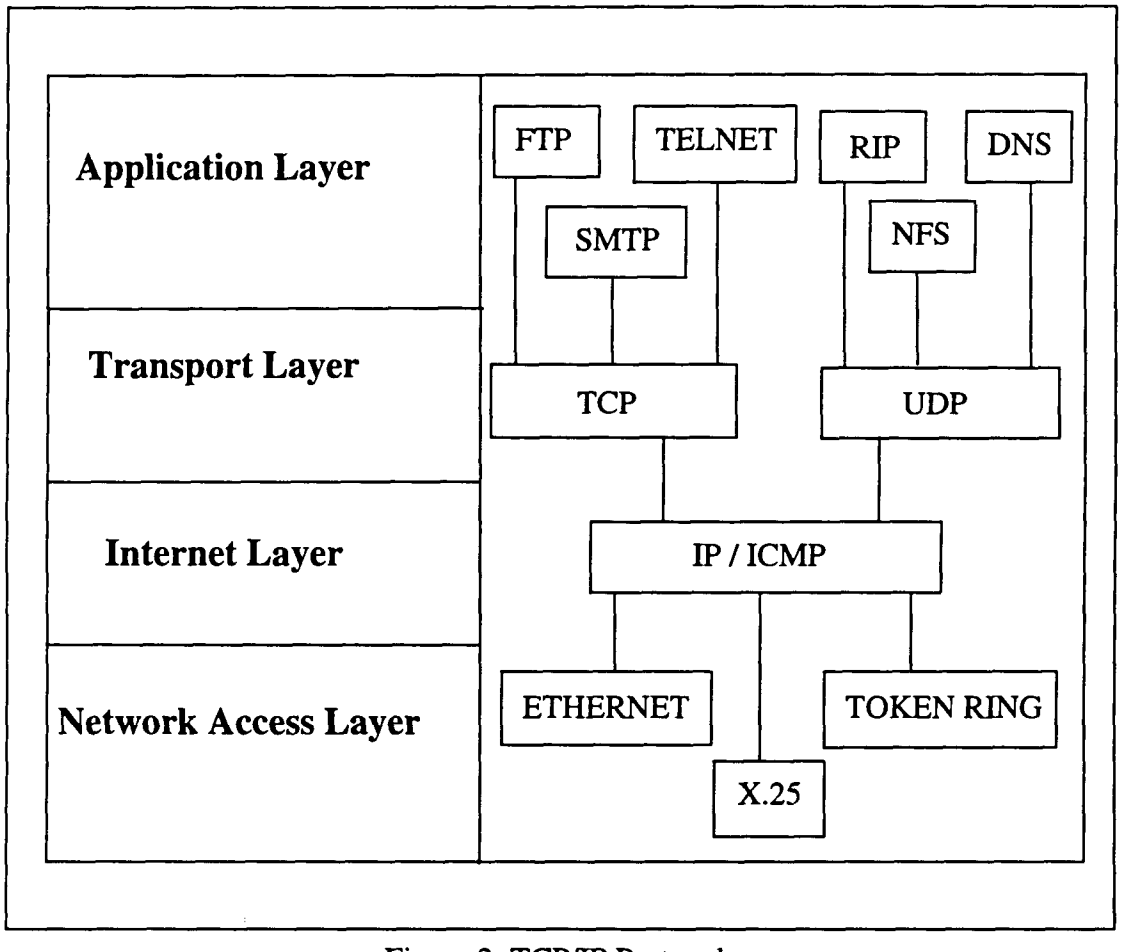

Figure 2: TCP/IP Protocols

The Network Access Layer of the TCP/IP protocol suite is the lowest layer where the device drivers are located. This layer must deliver data to the other devices on a directly attached network. The Network Access Layer must know the details of the underlying network to correctly format the data being transmitted. Examples of different protocols within the Network Access Layer are Ethernet, Token Ring, and X.25.

The Internet Layer is above the Network Access Layer. IP provides the basic packet delivery service on which TCP/IP networks are built. The protocols in this layer are responsible for routing data between networks and removing expired data packets. IP relies on protocols in other layers to establish the connection if they require connection-oriented service. IP also relies on protocols in the other layers to provide error detection and error recovery. Internet Layer protocols include the Internet Protocol (IP) and Internet Control Message Protocol (ICMP).

The Transport Layer is the host-to-host transport layer. The two most important protocols in the Transport Layer are Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). TCP ensures that data is delivered error free, in sequence, with no loss or duplication and is connection oriented. Basically, TCP is responsible for maintaining and monitoring the health and welfare of data exchange. UDP provides a low overhead, connectionless datagram delivery service. UDP sends bursts of data to another host rather than a stream of data.

The Application Layer is the top layer in the TCP/IP protocol architecture and is most widely known. Each service in this layer provides a different functionality or service for users. Some of the protocols within the Application Layer include File Transfer Protocol (FTP), Simple Mail Transfer Protocol (SMTP), Telecommunications Network Protocol (TELNET), Domain Name Service (DNS), Routing Information Protocol (RIP), Network File System (NFS), and Network Information Service (NIS).

2. Proxy Level Gateway

A proxy level gateway handles security via proxies at the Application Layer of the TCP/IP protocol stack. Another common name for this type of firewall is an application level gateway. A proxy firewall provides a high level of audit and security.

Every packet between the inside and outside networks passes through the firewall and goes up through each layer of the TCP/IP protocol suite. As shown in Figure 3, when the packets reach the Application Layer, a special purpose proxy intercepts the service request packets. The proxy checks its tables and denies or grants access to the service based on the

source's Internet address and the service being requested. If the service is denied, the packet is dropped, the event logged, and nothing further is done. If the service is granted, the event is logged, and the packets are passed on to the application of choice. The applications include FTP, SMTP, TELNET, DNS, RIP, NFS, or NIS.The fact that every service can control who uses it, is the primary advantage of proxy gateways and makes this type of firewall very secure.

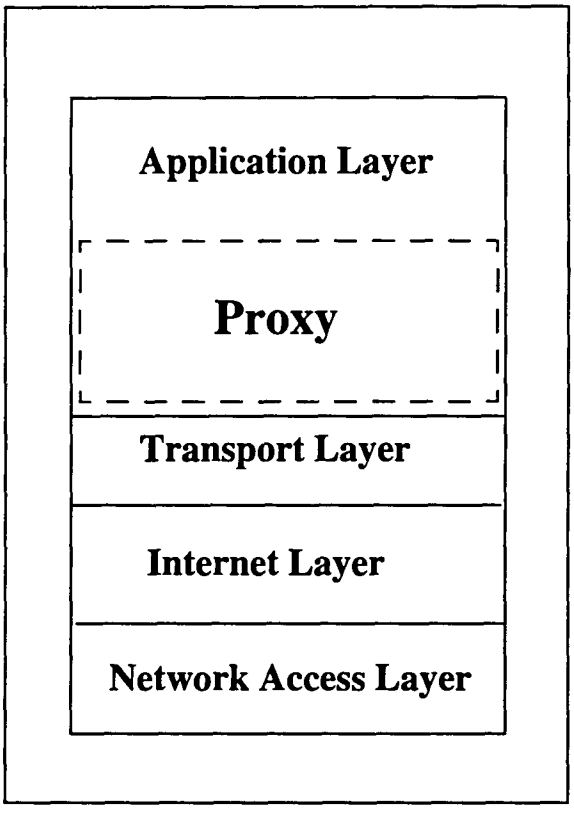

Figure 3: Proxy Level Gateway

The principal disadvantage of the proxy level gateway is the need for a specialized proxy for every service to be provided through the gateway. So, only the most important or most popular services can be supported. Another disadvantage is that this firewall is not transparent to the users. The users must know it exists and how to manage their work with it place. The use of each type of application is slightly different, depending on which one it is being used [CHES 94].

3. Packet Screening Filter

A packet screening filter is implemented at the IP level via screening rules. As shown in Figure 4, packets only need to go through the Network Access Layer and the Internet Layer of the TCP/IP protocol stack. At this point a general purpose filter is used to decide whether to deny or grant access. Packet filters work by analyzing packets based on their packet type (TCP, UDP, etc.), source IP address, destination IP address, and destination TCP/UDP port. Tables are configured for the packet screen filter that contain the information the filter uses to deny or grant access. The filter only controls which inside host's services are open, not which external hosts may access an internal host's service. The major disadvantage to packet screening is that the filters can be difficult to configure, modify, maintain, and test [CHAP 92].

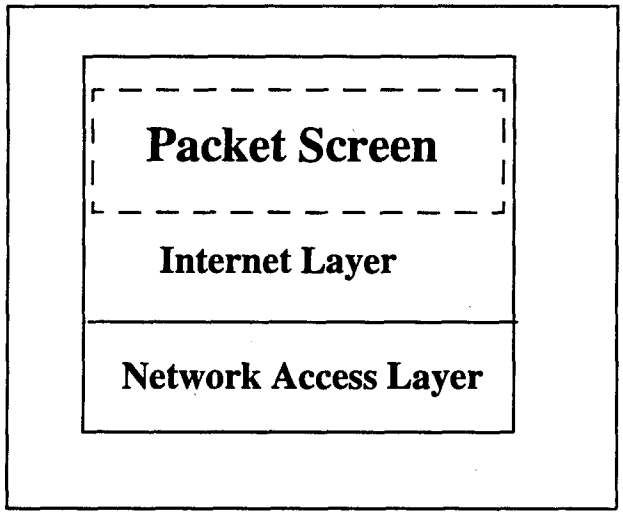

Figure 4: Packet Screening Filter

In general, the packet filter, once installed and configured, is transparent to the users, applications, and performance. This is the major advantage of this type of firewall. Only if a user or application is coming in from or going out from a machine which has been denied access, will the user notice the existence of the filter.

4. The Firewall at NPS

Based on the information above and the mission statement and political and technical environment at NPS, the decision was made to implement a packet filtering gateway with an external machine for added monitoring and logging for additional network security.

The primary reason the packet filter firewall was chosen was because the NPS campus is on the leading edge of research in computer technology and networking. The users and applications on campus can not be burdened with the limitations of the proxy level firewall. Also, NPS does not have enough staff to dedicate an individual to developing proxies when needed to be used on the firewall. As new applications are developed at NPS or elsewhere, a proxy would have to be developed for the firewall. The proxy gateways are considered to be a stronger line of defense, but the filtering gateway provides a level of compromise between security and availability more acceptable to the NPS environment and provides the flexibility required.

D. FIREWALL DESIGN

Instead of designing the firewall from the ground up, it was decided to use the firewall implemented at Texas A&M University (TAMU) and alter it as needed. TAMU developed this package of security tools as a result of over seven months of development and testing to protect an estimate of over 12,000 networked devices. The package contains three related sets of tools: drawbridge software package, netlog monitoring tools, and tiger programs [SAFF 93].

Figure 5 illustrates the overview of the filter and monitor implementation. The filter, running the drawbridge software, allows an arbitrary protocol filtering on a host by host basis. This provides a reasonable level of both security and flexibility for educational and research requirements. The monitor node, running the netlog monitoring tools, is placed outside the filter so that it can record connection attempts which are blocked by the filter. This is crucial to recognize intrusion attacks, but does not place the monitor itself at risk.

Both the filter and monitor will be placed in a controlled access machine room and the monitor configured for secure network access.

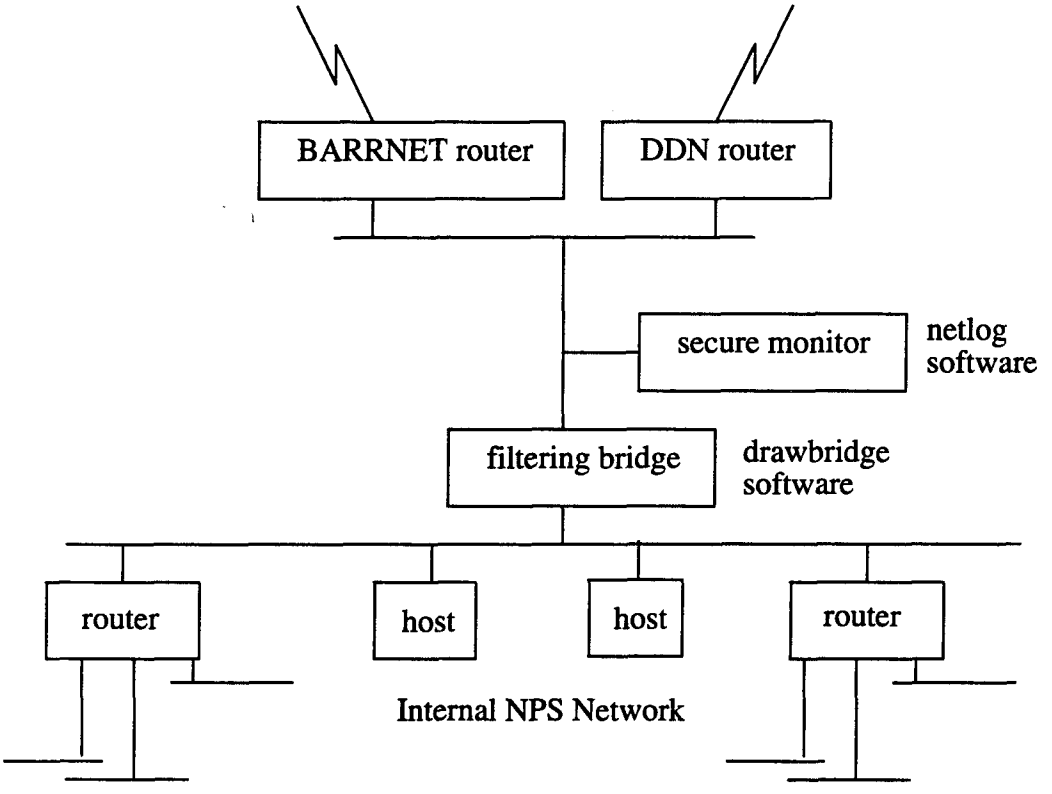

Figure 5: Firewall Overview

Initially, the filtering bridge will allow almost all packets through. There will only be a few restrictions placed on the filter. However, as the network security policy becomes solidified and supported by upper management at NPS, the filter can be used to grant Internet access only to those NPS hosts which are secure and are considered "cleared" by the network security committee. The security of each Unix host can be determined by packages such as the tiger programs.

1. Hardware Components

The filtering bridge running the drawbridge software is a 66MHz 486 personal computer (PC) with 4 megabytes of memory. It contains two SMC 8013 (Western Digital) ethernet cards. There is only a 20 megabyte hard drive in the PC because it only requires

about 7 megabytes for the MS-DOS 5.0 operating system and about 5 megabytes for the filter application.

The monitoring system is a 40MHz Sun SPARCStation 10 with 64 megabytes of memory. This system has two 424 megabyte internal hard drives. One of which will be exclusively used for the application software and log files.

2. Drawbridge Software Package

The drawbridge package contains a filter program that actually runs on the PC. This program is supported by a filter specification language and compiler that runs on a Sun Unix workstation (the same Sun workstation used for the monitoring software). The specification language and compiler are the components of the drawbridge package that addresses some of the recommendations made by [CHAP 92] with respects to the limitations of current filter implementations. They specifically address critical recommendations with respect to both functionality and ease of specification.

For both performance and configuration management, the filter tables are created on the support workstation, based on a powerful filter configuration language, and then transferred to the filter gateway (PC) dynamically and secured with a Data Encryption Standard (DES) authentication. The support workstation does the parsing of the configuration file, looking up addresses, and building the tables. The filter on the PC only needs to perform simple table lookups at run time.

Figure 6 illustrates the drawbridge's compilation and loading process. The filter source file is created on the monitoring workstation. This file contains the rule set used to perform the filtering. The rules are defined in terms of the host on the inside of the filter. No filtering is done based on the address of the host outside of the filter except on a global basis for reject and allow. It is assumed that an inside host will control which outside hosts are allowed access to its services. The filter only controls which inside host's services are open, not which outside hosts may access an inside host's services.

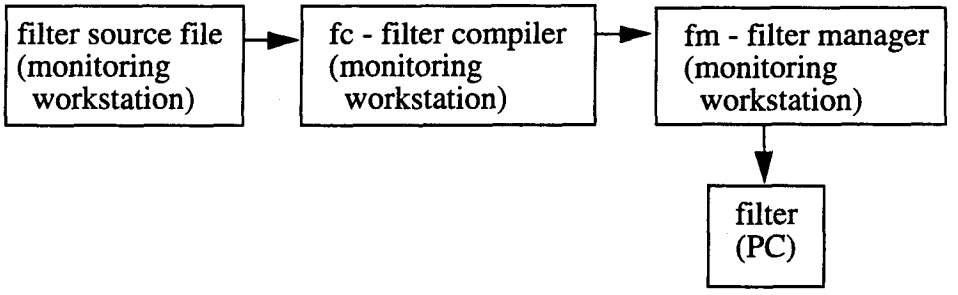

Figure 6: Drawbridge Compilation and Loading Process

Once this filter source file is created, it is compiled on the monitoring workstation using the fc, filter compiler. The compilation generates four output files for the rules which are then loaded into the file manager, fm. With the file manager the rule sets can be views and double checked. Once the rule sets are verified, the file manager is used to communicate with the packet filtering PC to upload the compiler's four output files. The filter on the PC runs the filter programs and is where the actual determination is made to grant or deny inbound and outbound packets based on the uploaded files created by the filter compiler.

The packet filtering PC is configured to run only the filter.exe software. The PC runs no other software and has no other device drivers installed. The autoexec.bat will be configured to automatically start the filter in case of power failures or reboots.

3. Netlog Monitoring Tools

In addition to running the fc and fm software, the goal of the monitoring workstation is to record security related network events by which intrusion attempts can be detected and tracked. The monitor itself must also be configured carefully, so it is secure.

The programs that can run on the monitor are described below.

a. tcplogger Netlog Tool

The tcplogger utility logs all TCP connection requests. This is accomplished by putting the network interface into promiscuous mode and reading all TCP connect request packets.

b. udplogger Netlog Tool

The udplogger tool logs the start of all UDP sessions. Like the tcplogger, this is accomplished by putting the network interface into promiscuous mode and reading all UDP packets. Only new sessions are logged.

c. icmplogger Netlog Tool

The icmplogger utility logs all ICMP sessions. Just like the tcplogger and udplogger, this is accomplished by the network interface being placed into promiscuous mode and reading all ICMP packets.

d. extract Netlog Tool

The extract utility displays records from a tcplogger or udplogger binary log file. The logs for both tcplogger or udplogger can be saved as ascii text or binary text. The binary text logs allow for more efficient logging. With the extract program, these binary logs are converted into ascii readable format.

e. netwatch Netlog Tool

The netwatch utility performs real time monitoring of an ethernet network, reporting any suspicious activity it may detect. Activities such as attempting to login to certain accounts, creating interactive shells via rsh, and unrecognized protocol commands are logged. The smtp driver attempts to detect interactive connections to the smtp port and reports these.

f. nstat Netlog Series of Tools

The nstat series of network monitor and analysis tools are useful for determining a network segment's overall load and utilization by service. This can be used for capacity planning, determining usage policies, and for spotting clandestine services.

This package includes three programs which are described below.

(1) nstat - A data collection program, counts packets and bytes going to and from all IP ports.

(2) nload - An awk program to collate data for plotting by xvgr.

(3) nsum - A perl program to print summary histogram.

4. Tiger Programs

The tiger scripts can be used on the individual Unix systems on the internal NPS network to locate security weaknesses. They search through a Unix system and report any elements of the system which may represent a security risk. Initially, the tiger programs will only be run on the monitoring system to secure that system.

The tiger scripts were written with ease of use in mind. They should be able to be run by people with possibly little Unix system administration background. Once an NPS policy is in place to "clear" any system that wants Internet access, the tiger programs can be run on a Unix system and the administrator of that system can simply provide a copy of the report that is generated when requesting the Internet access.

IV. FIREWALL VALIDATION

It is crucial the firewall work properly immediately after installation. Users and applications depend on the Internet services at NPS and that service can only be interrupted for a short period of time. The services that are expected to be blocked, must be blocked and the services that must be passed through, must do so. Also, the firewall must be able to handle the existing traffic between the NPS network and the Internet.

Before placing the firewall into an operational environment and possibly interfering with Internet services, a test network, as shown in Figure 7, was set up for a proof of principle.The proof of principle was used to validate the functionality and performance of the packet filtering PC.

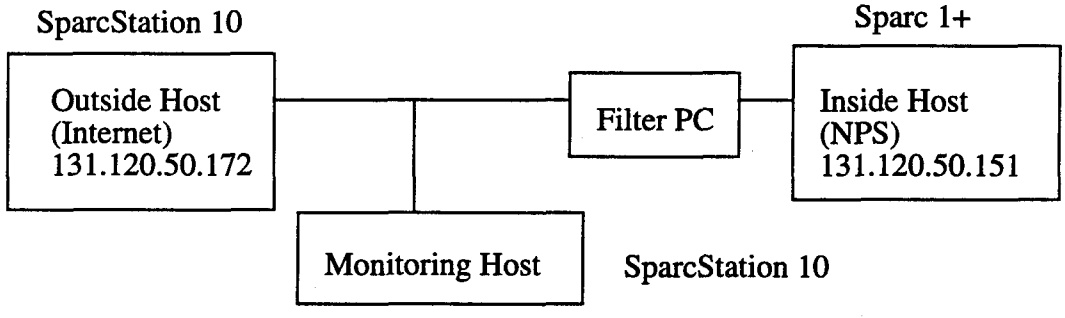

Figure 7: Test Network Configuration

The test network contains a network environment similar to the one being used at NPS. Three Sun Sparcs and a PC were used for this test network. One Sparc 1+ played the role of an inside host, a host attached to the NPS network. This Sparc was directly attached to an ethernet board in the PC. A second ethernet board in the packet filtering PC was connected to the SparcStation 10 which performed monitoring of network traffic. Then a SparcStation 10 was connected to the monitoring host and was considered the outside host, a host on the Internet. In the operational environment, the outside host would pass through

routers and gateways to gain access to NPS, but this test environment did not use routers and gateways.

A. FUNCTIONALITY ANALYSIS

1. Testing Drawbridge Filter Package

Since the goal of the firewall was to block unwanted packets, the first step was to look at all the different type of IP packets that the drawbridge can filter. This is basically every type of IP packet defined by vendors. Appendix B contains a large and all encompassing list of the common IP protocols supported by both TCP and UDP. It is impossible to actually filter all the common IP packets, however, there are a number of network services that are known to be buggy or are commonly used by hackers to gain access into networks. These are harmless network services, which become valuable tools in the search for weak points on systems, even when these services are operating exactly as they were designed. These network services are especially valuable when used together [FARM 93].

The security of the network at NPS will increase drastically by first identifying the services known to cause problems and then blocking the IP packets associated with those services. Table 5 contains a list of known network services that have been used by hackers to gain access to systems, including systems at NPS.

Network Service (alphabetical order)	Problem with Network Service
bootp	The bootp service provides information to the diskless clients on the network for booting purposes. If you can get the NIS domain-name you can then get the NIS maps, such as the password file.
DNS Domain Name Service	The domain name service is a problem because of the information it provides about hosts. This service could give an outsider a road map of how the hosts on the network are arranged.
finger	The finger command provides information about users on any host specified. Information such as account names, home directories, and the host the user last logged in from.

Table 5: Network Services With Known Problems

Network Service (alphabetical order)	Problem with Network Service
FTP File Transfer Protocol	Anonymous FTP sites can easily be misconfigured and are easy targets. Vulnerabilities are often a matter of incorrect ownership or permissions of key files or directories.
mount NFS Network File System	The NFS allows a convenient way to share file systems across a network through the mount network service. However, when misconfigured this utility can allow outsiders to mount your file systems and read and possibly write to them. This is an effective way to deposit trojan horses.
netwall	The netwall allows an emergency message to be displayed on all systems on the network. This utility can be used to broadcast a message to all users to change their password to a particular password. If the user doesn't question the broadcasted message and changes their password, an outsider has been given a way to access your system.
rlogin	The remote login has the potential of causing security problems if sniffers are used on the network. User keystrokes may be stolen, including passwords.
rpcinfo	This utility reports remote procedure call information. It can report if a host is running NIS, if it a NIS server or slave, if a diskless workstation is using the system to boot, or if the system is running NFS.
sendmail	Sendmail can provide information to the outside as to the current operating system being run and it's version number and the version of sendmail running. This can give hints as to how vulnerable it might be to attack.
showmount	The showmount command will list a system's NFS exports. These exports are directories that can be mounted by specified hosts on the network. If misconfigured, the directories can be exported to everyone and anyone can mount the directories.
telnet Telecommunications Network Protocol	The telnet service can be dangerous because of the potential use of sniffers on the network. Like the rlogin service, user keystrokes may be stolen, including passwords.
tftp Trivial File Transfer Protocol	The tftp program is similar to ftp but this program does not require passwords for authentication. If a host provides tftp without restricting the access, files can be read and written to anywhere on the system.

Table 5: Network Services With Known Problems

Network Service (alphabetical order)	Problem with Network Service
X Windows	If X windows is running and not protected properly, window displays can be captured or watched, user keystrokes may be stolen, programs executed remotely, etc.
ypxfr NIS Network Information Service	NIS is an insecure service that has almost no authentication between clients and servers. The ypxfr service is used for transferring the /etc/passwd file from the master server to the slave servers. If the NIS domain name is guessed, an outsider can get a copy of the /etc/passwd file.

Table 5: Network Services With Known Problems

Table 6 lists the IP protocol associated with each of the network services listed in Table 5. Some of the IP protocols should be blocked while others must not be blocked due to the extensive research and development done at NPS. Table 6 below indicates which ones were blocked and which were not.

The ftp, telnet, sendmail, DNS, finger, rlogin, and X windows services were not blocked, since these services are used extensively across campus. The DNS and sendmail services could be partially blocked. These services should theoretically be passing through to the particular NPS hosts that handle those services and not to all NPS hosts. However, the network at NPS is managed by many system administrators and each would have to indicate which hosts require DNS and sendmail services. This is a long process and outside the scope of this thesis. So, the DNS and sendmail services were not blocked.

Network Service	Port	IP Protocol	Block (yes / no)
ftp	20	ftp-data	no
ftp	21	ftp	no
telnet	23	telnet	no
rlogin	513 (TCP)	login	no
DNS	53	domain	no
bootp	67	bootp	yes

Table 6: IP Protocols Related to Network Services

Network Service	Port	IP Protocol	Block (yes / no)
netwall	533	netwall	yes
tftp	69	tftp	yes
finger	79	finger	no
ypxfr	111	sunrpc	yes
rpcinfo	111	sunrpc	yes
showmount	111	sunrpc	yes
mount	2049	nfsd	yes
sendmail	25	smtp	no
X windows	2000	openwin	no
X windows	6000 through 6xxx	X11 Window System	no

Table 6: IP Protocols Related to Network Services

The bootp, netwall, tftp, ypxfr, rpcinfo, showmount, and mount services were blocked. This means that none of these services will get through to an NPS host if a request is initiated from hosts outside of NPS. The decision was made not to block any outbound requests. Blocking the inbound services listed, creates enough of a barrier to keep most hackers coming in from the Internet from trying further to break into the NPS network.

Each service to be blocked was tested to validate that it is indeed blocked. The test network, Figure 7 on page 33, was used for this validation. Initially, the filtering PC was configured to allow all traffic through. Then one by one the services were disabled. After they were disabled, a test was made to validate that the service was disabled and by disabling the service, other services were not effected. This was an iterative process. The services, one by one, were tested from the workstation on the outside to access the workstation on the inside. Then the services were continued to be blocked and tested from the workstation on the inside to the workstation on the outside. Once an IP protocol was blocked, the corresponding service should no longer work inbound but all network services should work outbound.

The iterative process of testing was more efficiently done by using a remote terminal emulator (RTE). The RTE developed by Neal Nelson and Associates was used for the test. The purpose of the RTE is to imitate a user by generating a series of character transmissions to another computer. The RTE was used to capture the key strokes of an actual user performing each of the network services. The captured key strokes were logged into a script. Once the script was developed, it was played again and again for testing purposes.

The RTE script exercised each of the network services listed in Table 6 on page 36 with a few exceptions. The DNS service was impossible to configure properly within the test network, so it was eliminated from testing. The bootp and netwall services were also difficult to test. However, these two services were blocked on the PC filter and tests were run to verify no other services were effected.

Below is a list of the commands that were performed by the RTE script on the outside host, IP address 131.120.50.172, trying to reach the inside host, IP address 131.120.50.151. After testing from the outside host to the inside host, one test was performed from the inside host to the outside host. Note that the services not being blocked were included in the script to validate that they were not effected by the PC packet filter.

a. ftp Network Service

The following commands were performed to validate the two IP protocols, ftp-data and ftp (TCP ports 20 and 21), for the ftp network service:

outside% ftp 131.120.50.151

ftp> get /etc/passwd /tmp/passwd.inside

(login process...)

ftp> cd nttcp

ftp> get doit.sh

ftp> quit

b. telnet Network Service

The following commands were performed to validate that the IP protocol, telnet (TCP port 23), for the telnet network service:

outside% telnet 131.120.50.151

(login process...)

inside% cat /etc/passwd

inside% exit

c. rlogin Network Service

The following commands were performed to validate the IP protocol, login (TCP port 513), for the rlogin network service:

outside% rlogin altair.cc.nps.navy.mil -l jody

(login process...)

inside% cat /etc/passwd

inside% exit

d. tftp Network Service

The following commands were performed to validate that the IP protocol, tftp (UDP port 69), for the tftp network service:

outside% tftp

tftp> connect 131.120.50.151

tftp> get /etc/passwd /tmp/passwd.inside

tftp> quit

e. finger Network Service

The following command was performed to validate that the IP protocol, finger (TCP port 79), for the finger network service:

outside% finger jody@131.120.50.151

f. ypxfr Network Service

The following command was performed to validate that the IP protocol, sunrpc (TCP / UDP ports 111), for the ypxfr network service:

outside% /usr/etc/yp/ypxfr -b -c -f -d TestDomain -h 131.120.50.151 passwd.byname

g. rpcinfo Network Service

The following command was performed to validate that the IP protocol, sunrpc (TCP / UDP ports 111), for the rpcinfo network service:

outside% rpcinfo -p 131.120.50.151

h. showmount Network Service

The following command was performed to validate that the IP protocol, sunrpc (TCP ports 111), for the showmount network service:

outside% showmount -e 131.120.50.151

i. mount Network Service

The following commands were performed to validate the IP protocol, nfsd (UDP port 2049) and sunrpc (TCP / UDP port 111), for the mount network service:

outside% mount 131.120.50.151:/home /mnt

outside% cd /mnt

outside% ls

j. sendmail Network Service

The following command was performed to validate that the IP protocol, smtp (TCP port 25), for the sendmail network service:

outside% telnet 131.120.50.151 25

k. X windows Network Service

The following commands were performed to validate the IP protocol, openwin (TCP port 2000), for the X window network service:

outside% xhost 131.120.50.151

outside% telnet 131.120.50.151

inside% setenv DISPLAY 131.120.50.172:0.0

inside% /usr/openwin/bin/filemgr

After the RTE script was developed using the commands described above, the test runs listed below were executed and the results documented. Before each test was executed, the filter was reconfigured to block or not block the IP packets indicated.

l. Test Run 1 - No Packets Blocked

This test was run from the machine on the outside (IP address 131.120.50.172), accessing the machine on the inside (IP address 131.120.50.151) of the PC packet filter. No packets were blocked by the PC filter.

During this test run, all the commands in the RTE script completed successfully with no error messages.

m. Test Run 2 - Block bootp and netwall Packets

This test was run from the machine on the outside (IP address 131.120.50.172), accessing the machine on the inside (IP address 131.120.50.151) of the PC packet filter. The bootp (UDP ports 67 and 68) and netwall (UDP port 533) IP packets were blocked by the PC filter.

During this test run, no network services executed by the RTE script failed. This validated that no services were impacted by blocking the bootp or netwall packets.

n. Test Run 3 - Block Only tftp Packets

This test was run from the machine on the outside (IP address 131.120.50.172), accessing the machine on the inside (IP address 131.120.50.151) of the PC packet filter. Only the tftp (UDP port 69) IP packets were blocked by the PC filter.

During this test run, the tftp network service failed causing error messages. No other network services failed.

o. Test Run 4 - Block Only sunrpc Packets

This test was run from the machine on the outside (IP address 131.120.50.172), accessing the machine on the inside (IP address 131.120.50.151) of the PC packet filter. Only the sunrpc (TCP and UDP port 111) IP packets were blocked by the PC filter.

During this test run, the ypxfr, rpcinfo, showmount, and mount network services failed causing error messages. No other network services failed.

p. Test Run 5 - Block Only nfsd Packets

This test was run from the machine on the outside (IP address 131.120.50.172), accessing the machine on the inside (IP address 131.120.50.151) of the PC packet filter. Only the nfsd (UDP port 2049) IP packets were blocked by the PC filter.

During this test run, the mount network service failed causing error messages. No other network services will fail.

q. Test Run 6 - Block All Packets Listed in Tests 2 to 5 (Inbound)

This test was run from the machine on the outside (IP address 131.120.50.172), accessing the machine on the inside (IP address 131.120.50.151) of the PC packet filter. The bootp (UDP ports 67 and 68), netwall (UDP port 533), tftp (UDP port 69), sunrpc (TCP and UDP port 111), and nfsd (UDP port 2049) IP packets were blocked by the PC filter.

Table 7 lists the network services that failed during this test run and caused error messages and the network services which did not fail. The table below does not list the DNS, bootp, or netwall network services because those services are not directly tested with the RTE script.

Network Service	Expected Result
ftp	no error messages
telnet	no error messages
rlogin	no error message
sendmail	no error messages
tftp	error message
finger	no error message
ypxfr	error message
rpcinfo	error message
showmount	error message
mount	error message
X windows	no error message

Table 7: Test Run 9 Expected Results

r. Test Run 7 - Block All Packets Listed in Tests 2 to 5 (Outbound)

This test was run from the machine on the inside (IP address 131.120.50.151), accessing the machine on the outside (IP address 131.120.50.172) of the PC packet filter. No packets will be blocked by the PC filter.

During this test run, all the commands completed successfully with no error messages.

The functionality testing demonstrated that the TAMU packet filter operated as required, blocking only those packets specified, for all seven test runs.

B. PERFORMANCE ANALYSIS

An issue of great concern was whether or not the packet filtering firewall on the 66 MHz 486 PC has the capacity to handle the day-to-day traffic currently being exchanged between the Internet and the NPS network, when it is placed into it's final position as shown in Figure 5 on page 27.

First, the traffic between the NPS network and the Internet had to be measured. This involved looking at the routers to DDN and to BARRNET and determining the traffic sent and received by those routers.

Second, the performance testing of the PC packet filter had to be determined. The performance testing involved two different configurations. Figure 8 on page 44 illustrates these two configurations. One configuration was with the packet filtering PC in place and operational and the second configuration was with no packet filtering PC in place.

The performance of the PC was analyzed from three different perspectives within the two configurations illustrated in Figure 8. The first approach was to see if the PC could handle TCP/IP packets pumped over the network using the New Test Transmission Control Protocol (*nttcp*). This utility eliminates any overhead produced by the applications, such as ftp or telnet, and determines the raw network speed. The second approach looked at the throughput information returned after an ftp command was issued. This information was returned in kilobytes per second and allowed a view of the performance from the user's

point of view. Finally, the third approach was using the RTE to determine the performance degradation from 1 to 16 users. The RTE script performed the commands ftp and telnet which use TCP/IP for 1 through 16 users (in increments of 1) and measured the time it took, in seconds, to receive a response to those commands.

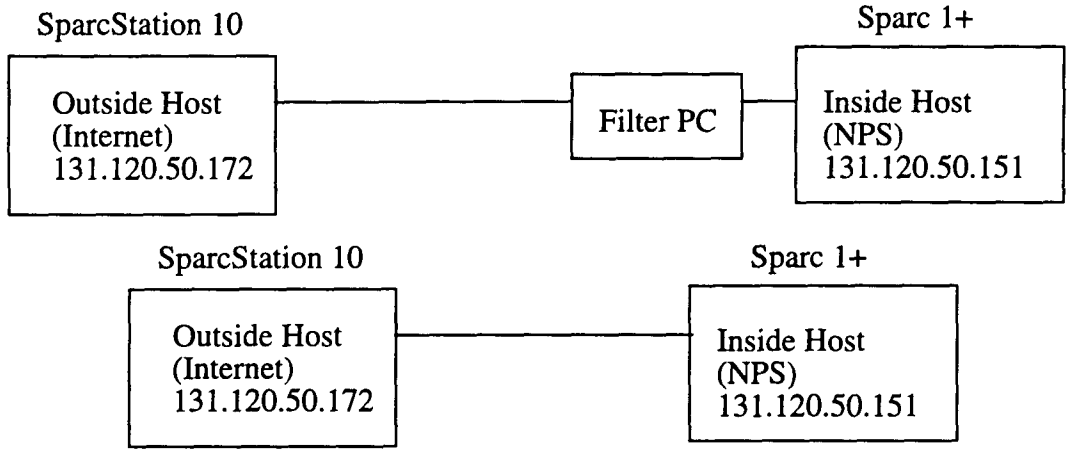

Figure 8: Configurations for Performance Testing

1. Current Internet Traffic

The Naval Postgraduate School currently has two different access points to the Internet. One connection is via the DDN router and the second connection is through the BARRNET router, see Figure 9 on page 45. The DDN router has a 56 Kbps connection to the outside network and the BARRNET router has a T1 (1.544 Mbps) connection to the outside network. However, the BARRNET router's T1 line is shared with at least four other organizations. So, the theoretical capacity to the Internet is up to 2 Mbps. However, this capacity is not fully utilized because it is shared.

What part of the 2Mbps is being utilized? To answer this question, a Hewlett Packard (HP), Series 386, Network Advisor was connected to the network and the two Internet connections were monitored. Two types of monitoring were done. The first was to monitor the network traffic to and from the two routers over a week period. The second was to monitor the network traffic to and from the two routers in 15 minute intervals over a 14 hour period in the middle of the week. The one week period provided a good overall data

rate to and from the DDN and BARRNET routers and the one day monitoring period provided the peaks and valleys.

The HP analyzer has the ability to report the total number of bytes sent and received by a particular node on the network. When monitoring was started the start time was recorded (in hours, minutes, and seconds). When the monitoring period was complete, the monitor was stopped and the stop time recorded along with the bytes sent and received by the DDN and BARRNET routers. Once the HP analyzer was restarted, the bytes sent and received were automatically reset to zero prior to the next monitoring period.

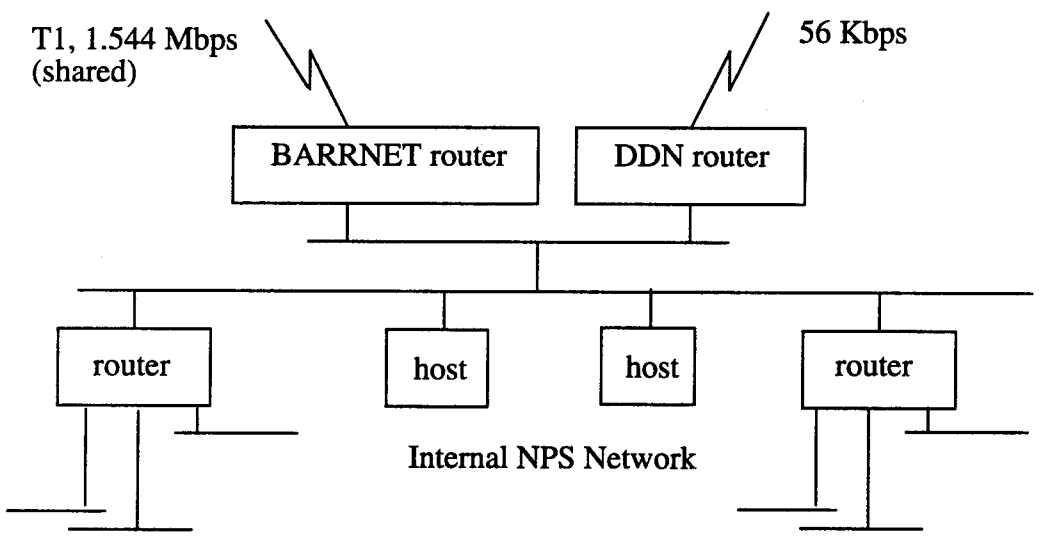

Figure 9: Current Internet Traffic

Appendix C contains the raw data results in bytes from both the monitoring for the one week period and the one day period. In addition to the raw data results, the packet overhead was determined. Samples were taken of the total number of frames sent and received and the total number of bytes sent and received. Then the overhead was calculated by using the equation below:

$$frameoverhead = 1.0 - ((total) / (total) + (frames \times 26))$$

The constant 26 is the number of bytes of overhead per frame. The *total* and *frames* variables were derived from the following:

total = total number of bytes sent and received

frames = total number of frames sent and received

The frame overhead was determined to be on average 6.279%.

Using the raw data from Appendix C and the frame overhead the actual data rate was determined using the following equation:

$$datarate = (((totbytes \times 8) / (elapsedtime)) \times (1 + overhead)) / 1000)$$

The *datarate* determined above is in the form of kilobits per second. The constant 1000 converts the bits per second to kilobits per second. The other constant, 8, was used to convert the total number of bytes (*totbytes*) into bits. The *overhead* variable is equal to the 6.279% (.06279) overhead calculated above and *totbytes* and *elapsedtime* variables were derived from the following:

totbytes = total number of bytes sent and received

elapsedtime = seconds between the start and stop of each measurement interval

The *datarate* was calculated for each measured time intervals for the one week monitoring period and the one day monitoring period. These calculations are included in Appendix C. From the datarates in Appendix C an average *datarate* was calculated. The one week monitoring period showed that the actual traffic to and from the DDN and BARRNET routers was on average 306.81 kilobits per second. The one day monitoring period showed that the actual traffic to and from the DDN and BARRNET routers was on average 438.17 kilobits per second.

These results indicate that any firewall placed between the Internet and NPS network must be able to handle 438.17 kilobits per second on the average. If the firewall cannot sustain that data rate, the firewall would become a bottleneck and users would notice slower performance with their Internet accesses with the firewall in place and operational. If the firewall can sustain the data rate currently being generated by the DDN and BARRNET routers, then the firewall would not cause any impact on Internet performance.

2. New Test Transmission Control Protocol

The tool used for the network performance analysis was the New Test TCP *(nttcp)* which uses Test TCP *(ttcp)*. The *nttcp* utility is the basic tool for determining measured throughput over any physical network media. The original *ttcp* utility was developed by the U. S. Army's Ballistic Research Lab (BRL) which is now the U. S. Army's Research Lab (ARL) and is considered one of the default network performance benchmarks.

The *nttcp* program tests TCP and UDP performance by timing the transmission and reception of data between two systems using the UDP or TCP protocols. It differs from common "blast" tests, which tend to measure the remote Internet daemon (*inetd*) software as much as the network performance and usually does not allow measurements at the remote end of a UDP transmission.

This tool provides many tunable options including the dynamic changing the TCP/IP window size during the throughput test. The different options are as follows:

-t Transmit mode.

-r Receive mode.

-u Use UDP instead of TCP.

-n Number of source buffers transmitted.

-l Length of memory buffers in bytes.

-w TCP/IP window size in k bytes.

-p Port number to send to or listen on.

For testing, the transmitter should be started with -t after the receiver has been started with -r. For testing various window sizes, *nttcp* allows a -w option which permits the user to specify the desired TCP/IP window size.

For this performance analysis, a window size of 4096 was used. This value is the default ethernet window size on the Sun workstations. With the window size fixed, data was transmitted between the two systems. Two other variables were used and varied, the number of source buffers transmitted and the length of the memory buffers, in bytes.

Changing these two variables, -n and -l, is equivalent to passing different file sizes from one system to another.

The shell scripts along with the *nttcp* program are in Appendix D. The shell scripts *doit.sh* and *ttest.sh* were written by personnel at the U. S. Army Research Lab (ARL) and modified to fit this research. These scripts were designed to be used with the program *nttcp*. The first script, *doit.sh*, calls the shell script *ttest.sh*. The script *doit.sh* provides the various combinations of data sizes to be transferred along with starting and stopping times of each run. The *doit.sh* script runs through six iterations of identical data sets. The shell script *ttest.sh*, provides the calls to the program *nttcp*. Using the data length and number of packets specified in the shell script *doit.sh, ttest.sh* makes the calls to *nttcp* with the constant window size of 4. This combination of amount of data transferred and the number of test runs provides a total of 48 measured data transfers during a single run. Amount of data transferred (8 sizes) * number of test runs (6 runs) * = 48 measured data transfers. From these measured data transfers, an average megabytes per second can be calculated.

The test network illustrated in Figure 7 on page 33 was modified for the performance testing using *nttcp*. The modification resulted in two test networks for the *nttcp* tests and these are shown in Figure 8 on page 44. The following tests were run using the *nttcp* utility.

a. Test Run 1 - Without Packet Filter PC in Place

The first test run used the shell scripts in Appendix D. The *nttcp* test was initiated on the outside host. The inside host transmitted the data and the inside host received the data. No packet filtering PC was in place. The inside and outside hosts were directly connected to one another.

The actual raw results from this test run are contained in Appendix E. The average data rate of the 48 measured data transfers was 5.21 megabits per second.

b. Test Run 2 - Packet Filter PC in Place (Outbound)

The second test run used the shell scripts in Appendix D. The *nttcp* test was initiated on the inside host. The inside host transmitted the data and the outside host received the data. For this test, the packet filtering PC was in place. The filtering PC was

placed between the hosts and configured to pass all packets through, no packets were blocked.

The actual raw results from this test run are contained in Appendix E. The average data rate of the 48 measured data transfers was .645 megabits per second.

c. Test Run 3 - Packet Filter PC in Place (Inbound)

The third test run also used the shell scripts in Appendix D. The *nttcp* test was initiated on the outside host. The outside host transmitted the data and the inside host received the data. For this test, the packet filtering PC was in place. The filtering PC was placed between the hosts and configured to pass all packets through, no packets were blocked.

The actual raw results from this test run are contained in Appendix E. The average data rate of the 48 measured data transfers was .595 megabits per second.

3. ftp Test

The ftp command returns the kilobytes per seconds it takes to complete the put or the get command. This is a good indication of the performance of the connection between two hosts. This series of tests performed the ftp commands put and get using a 2 megabyte file to the /dev/null device. The kilobytes per second returned from the commands were recorded for each of the test runs listed below.

a. Test Run 1 - Without Packet Filter PC in Place

The first test run was executed on the outside host. Using ftp, a put command and then a get command were executed by hand to the /dev/null device. The inside and outside hosts were directly connected to one another.

This test run returned on average 1.2E02 kilobytes per second for the put command and 1.2E02 kilobytes per second for the get command.

b. Test Run 2 - Packet Filter PC in Place (Outbound)

The second test run was executed on the inside host. Using ftp, a put command and then a get command were executed by hand to the /dev/null device. The filtering PC

was placed between the hosts and configured to pass all packets through, no packets were blocked.

This test run returned on average 58.8 kilobytes per second for the put command and 55.4 kilobytes per second for the get command.

c. Test Run 3 - Packet Filter PC in Place (Inbound)

The third test run was executed on the outside host. Using ftp, a put command and then a get command were executed by hand to the /dev/null device. The filtering PC was placed between the hosts and configured to pass all packets through, no packets were blocked.

The test run returned on average 55 kilobytes per second for the put command and 57.8 kilobytes per second for the get command.

4. Remote Terminal Emulator Performance Test, 1 to 16 Users

The RTE is an ideal tool to use for performance testing. User commands were captured into the RTE scripts. Once the script were developed, they were played back and the RTE recorded the responses from the other system and tracked the amount of time the other system took to respond to a command. The RTE was used to replay the scripts for more than one user. For the purpose of this testing, the RTE replayed the script for one user up through 16 users, in increments of one user at a time.

The RTE scripts was designed to perform the following commands:

Establish an ftp connection.

Using ftp, put a 2 megabyte to /dev/null.

Using ftp, get a 2 megabyte file to /dev/null.

Establish a telnet connection.

Using telnet, cat a 2 megabyte file.

Measurements were taken for each of the commands listed above. For the ftp command, three measurements were made. The first measurement was how long it took to establish the ftp connection with the other system. Once the ftp connection was established, the second timing measurement was how long it took to put a 2 megabyte file and the third

timing measurement was how long it took to get a 2 megabyte file. The timing measurement began at the time the return key is entered for the get command and stopped once the "ftp>" prompt was returned from the other host.

For the telnet command, there were also two measurements made. The first measurement was how long it took to establish the telnet connection and the second measurement was how long it took to cat a 2 megabyte file to the screen.

After the RTE script was developed using the commands and timing measurements described above, the test runs listed below were executed. Like with the *nttcp* tests, the test network shown in Figure 7 on page 33 was modified for this performance testing. The modification resulted in the two test networks illustrated in Figure 8 on page 44.

a. Test Run 1 - Without Packet Filter PC in Place, 1 to 16 Users

The first test run was executed on the outside host accessing the inside host, using the RTE script described above with the timing measurements in place, for 1 to 16 users (incrementing by 1 user at a time). The inside and outside hosts were directly connected to one another.

b. Test Run 2 - Packet Filter PC in Place, 1 to 16 Users

The third test run was executed on the outside host accessing the inside host, using the RTE script described above with the timing measurements in place, for 1 to 16 users (incrementing by 1 user at a time). The filtering PC was placed between the hosts and configured to pass all packets through, no packets were blocked.

The results from both test runs above were combined. This resulted in one graph for each of the measure intervals, ftp, put, get, telnet, and cat. Figure 10 through Figure 14 below contain the results (the median) for both test runs described above for 1 to 16 users.

Figure 10 on page 52 illustrates the median time, in seconds, it took to establish the ftp connection for 1 through 16 users. There was relatively little difference between this measurement interval with or without the TAMU PC packet filter. This is expected, since the ftp establishment is a relative short event going across the network

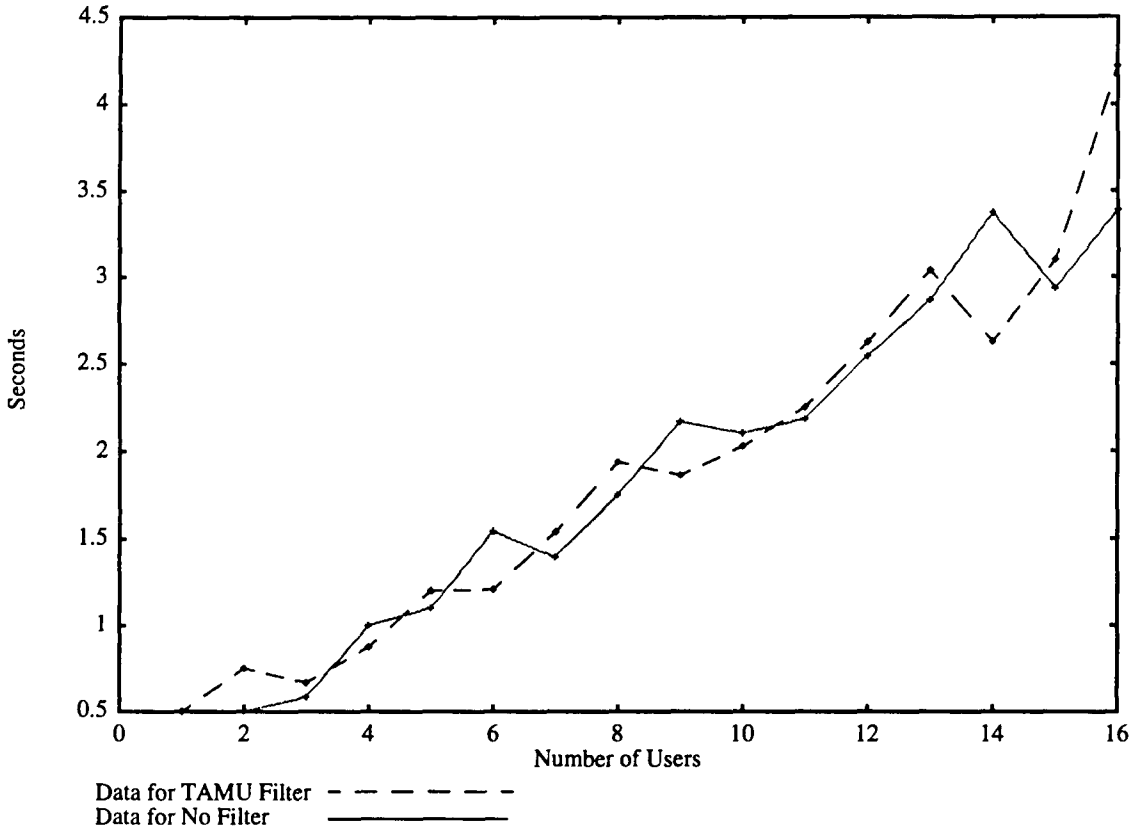

Figure 10: RTE Test 1 to 16 Users, "ftp" Command

Figure 11 on page 53 illustrates the median time, in seconds, it took to put a 2 megabyte file with the ftp utility. This figure shows that without the PC packet filter, the amount of time to put the 2 megabyte file, as the number of users increases, was basically the same. However, when the TAMU packet filter was installed, the response time in seconds from 1 to 16 users became quite a bit longer. One interesting note was that the peaks and valleys in the graph are basically in the same locations however, the TAMU filter caused them to be magnified. These are disturbing results, but predictable.

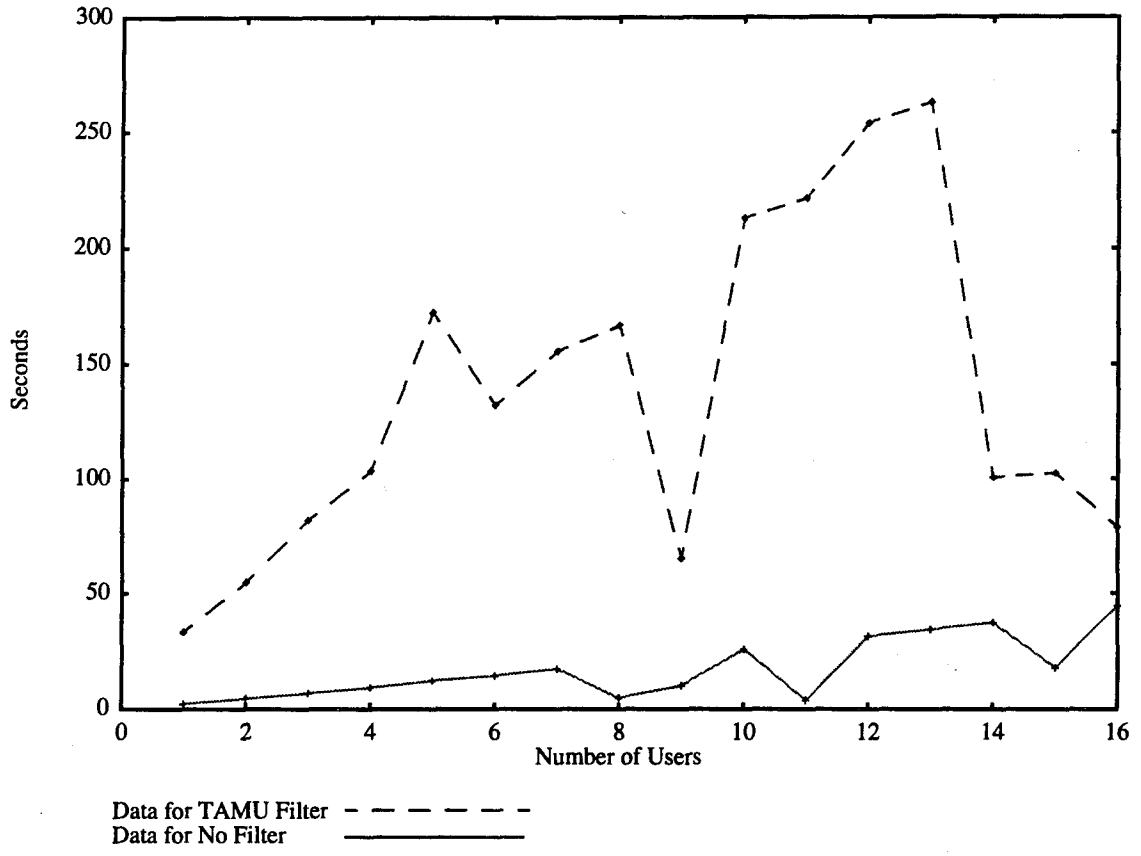

Figure 11: RTE Test 1 to 16 Users, "put" Command

Figure 12 on page 54 illustrates the median time, in seconds, it took to get a 2 megabyte file with the ftp utility. This figure shows that without the PC packet filter, the amount of time to get the 2 megabyte file, as the number of users increases, was basically the same just like the put command discussed above. However, when the TAMU packet filter was installed, the response time in seconds from 1 to 16 users became quite a bit longer. One interesting note is that the peaks and valleys in the graph are basically in the same locations, however, the TAMU filter causes them to be magnified. The put and get commands are closely aligned. These are expected results.

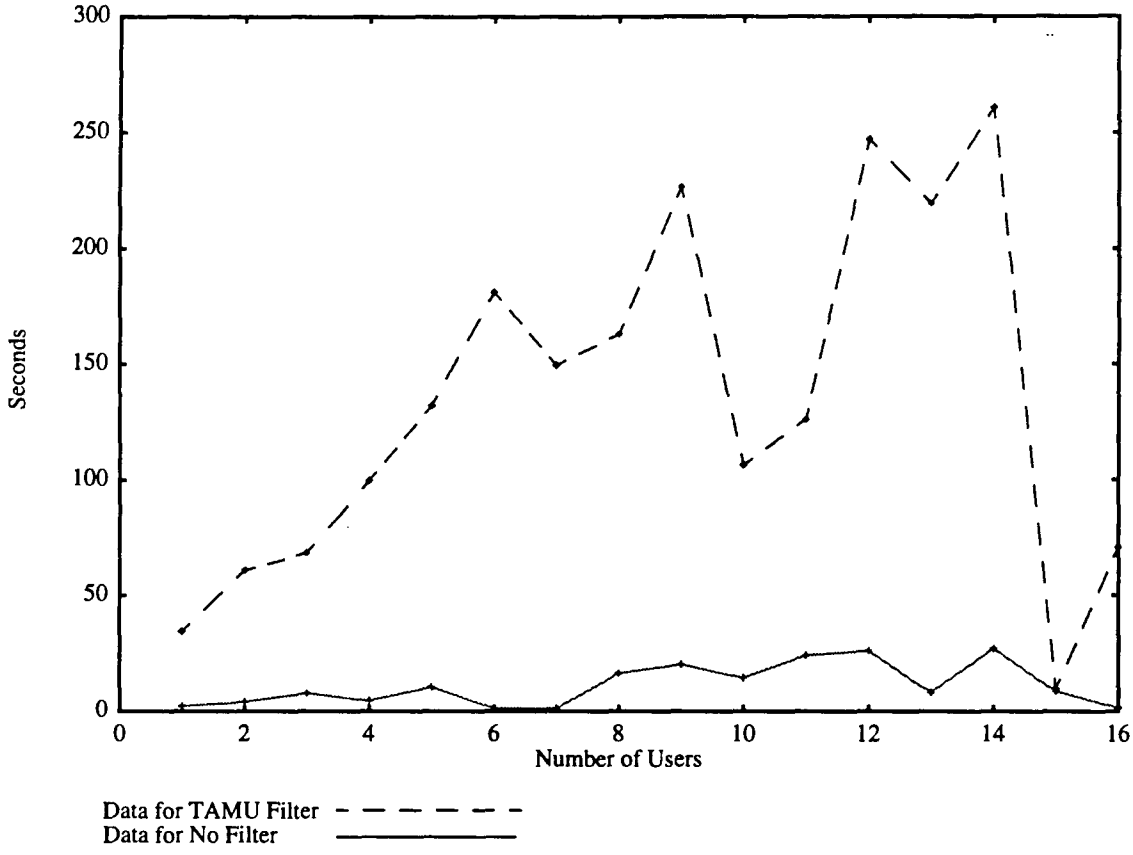

Figure 12: RTE Test 1 to 16 Users, "get" Command

Figure 13 on page 55 illustrates the median time, in seconds, it took to establish the telnet connection for 1 through 16 users. There was relatively little difference between the measurement intervals with or without the TAMU PC packet filter. This was expected, since the telnet establishment, like the ftp establishment, is a relative short event going across the network.

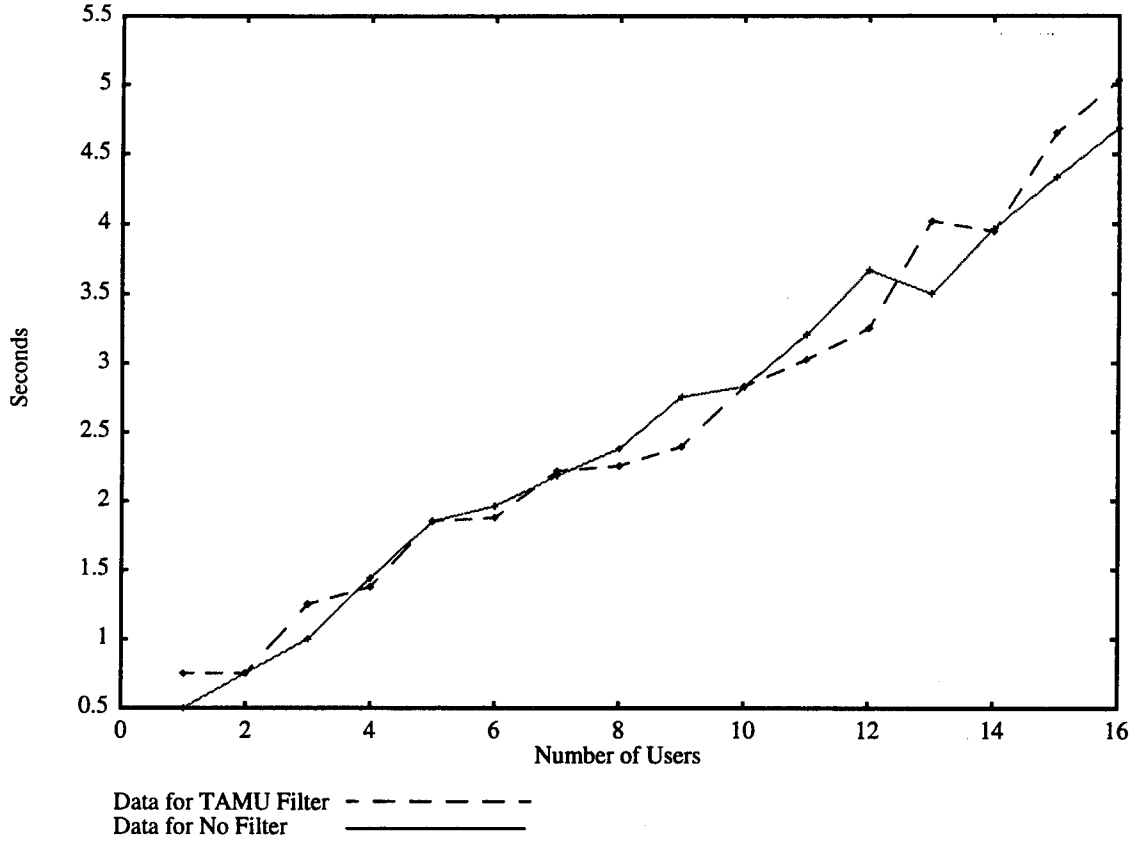

Figure 13: RTE Test 1 to 16 Users, "telnet" Command

Figure 14 on page 56 illustrates the median time, in seconds, it took to cat a 2 megabyte file after logging into the remote computer with telnet. This figure shows that the PC packet filter actually improved the performance of the cat command compared to having no filter in place. This can be explained. The cat command is bidirectional. With no filter, the number of collisions increases and causes a poorer response time. The filter adds enough of a delay to reduce the number of collisions. These results show that the slopes map closely for both sets of data, which is desirable.

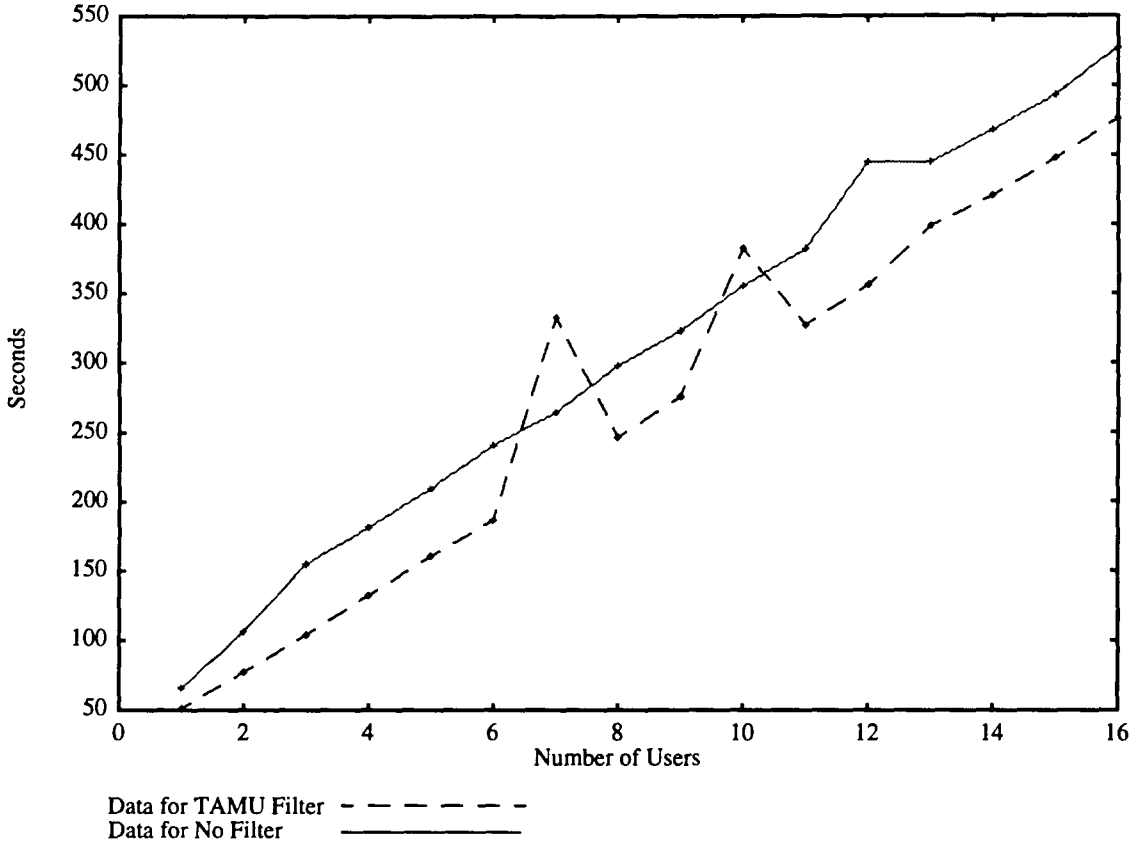

Figure 14: RTE Test 1 to 16 Users, "cat" Command

C. SUMMARY

The results of the functionality testing were favorable. The TAMU PC packet filtering firewall functioned as it should. It is easy to configure and change on a moment's notice. These are both good qualities.

The results of the performance testing were not expected. The slow data rate experienced when the firewall was installed is a potential bottleneck. Without the filter installed, the throughput was 5.21 megabits per second. With the PC packet filter, the throughput dropped to about .6 megabits per second. Even though the average traffic of the Internet connections is currently 438.17 kilobits per second there were peaks of over 1,000 kilobits per second and that could over load the firewall. Also, the RTE test results indicated a notable difference in the user's response time with the PC packet filtering

firewall installed. How the user perceives the performance of the network is extremely important. The RTE results indicated that the users will perceive a noted decrease in network performance, especially as the number of users accessing the Internet across the firewall increases.

V. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

There is no dispute that the connectivity NPS has to the Internet is vital to the mission of the school. This thesis explored how the school can continue to benefit from the Internet connection while protecting the network assets from the threats the Internet connection creates. The issues and questions that were covered by this thesis work are listed below in addition to the way each issue was addressed by this thesis.

What is the operational definition of network security in an environment like NPS? The actual network security policy at NPS is no different from that established by NCSC; however, it is implementation where the difference lies. The first step was to determine the minimum requirements for a network security policy at NPS. Since NPS is a government organization, there are DoD Directives that must be followed. A formal risk assessment procedure was used to determine the minimum evaluation class required for automated information systems, including networks, based on the sensitivity of the information present and on the clearance of its users. The risk assessment determined that NPS is required to provide B1 level of security for the NPS network.

What network security model best suits NPS? Since it is impossible to achieve B1 level security on every host on the NPS network in the immediate future, the approach of perimeter security was taken by this thesis with the use of an Internet firewall. This approach provides a higher level of network security than what was previously in place while serving the varied needs of the school's academic staff and mission.

Would a network firewall be effective in eliminating or reducing the current threats while sustaining the users functionality? This thesis demonstrated that network services which have been documented to have vulnerabilities can be blocked from the Internet while maintaining the functionality desired by the users.

If a network firewall would be effective, what would be the most efficient and most cost effective implementation? The most efficient and cost effective firewall

implementation was public domain software available free of charge from the Internet. The software from Texas A&M University (TAMU) was designed for an academic environment, easy to use, and implemented on a 486 PC. So the cost was minimal.

B. APPLICATION OF RESEARCH

The research performed for this thesis is applicable to any individual that is responsible for maintaining the security of systems or of networks.

1. System Administrators

System administrators have a difficult job. They have broad responsiblities for a wide variety of computers. To list a few of the administrator's duties, they have to know about operating systems, applications, programming, system accounting, account administration, backups, restoring data, and security. Just within the area of security, there are the subcategories of physical security, data security, system security, and network security. So the administrator's job is not easy.

The Internet firewall analyzed by this thesis is a compliment to what the system administrators already do on a daily basis. It helps give an added layer of protection to the systems each of the administrators are responsible for maintaining. Some administrators do not have the experience and/or the time to effectively handle system security. Therefore, the firewall would provide a minimum layer of security to all systems attached to the NPS network, no matter how well the systems themselves are maintained.

The results of this thesis can help system administrators understand what they can expect in the functionality and performance of the firewall when it is put into operation.

2. Network Administrators

Network administrators are basically responsible for the overall welfare of the network. If an intruder is coming in from the outside, the network administrator has more control if there is a central location from which they can block inbound access to the NPS network. The firewall provides central control of all inbound and outbound Internet traffic.

The results of this thesis demonstrate that installing a firewall between the NPS network and the Internet would make a network administrator's duties easier by providing a central location to tighten access as required. This thesis can help the network administrator understand what to expect in the area of throughput with an installed firewall.

3. Automation Data Processing (ADP) Security Officers (ADPSO)

The ADPSO has the responsibility for ensuring security on all automated information systems at NPS. This is an extremely difficult and daunting task. The firewall research accomplished by this thesis will provide the ADPSO with the ability to enforce a security policy on the network.

Initially, the firewall can be configured to allow all NPS hosts access to the Internet. However, the firewall can later be configured to only allow authorized NPS hosts to access the Internet. This will be effective when tighter security is enforced on a system by system basis at NPS. At that time, if a host desires Internet access the administrator must demonstrate that the host is secure before that host is added to the firewall's access tables.

4. Management

The management at NPS must be concerned about access to NPS hosts via the Internet since, historically, hackers will typically target military computer sites.

The results of this thesis illustrate that the well know and well publicized holes in the network utilities can be blocked to improve computer and network security overall at NPS. The firewall is an effective measure to ensure a blanket level of security across the NPS network.

5. Research Scientists

The research community can gain insight on what additional research needs to be accomplished in the area of firewalls. This thesis shows the basic functionality of the firewall is sound and that firewalls are valuable assets towards enhancing security.

However, firewall throughput does leave open questions that should be researched further. This thesis illustrates that performance is a major issue surrounding firewall technology.

C. RECOMMENDATIONS

There are, at a minimum, four areas of research that can be pursued from this thesis. These areas are performance, functionality, inclusion of dial-in access, and formal verification of the firewall software.

The lack of performance of the firewall identified in this thesis suggests further research into the cause of the bottleneck. The software could possibly be the cause or it could be that a PC does not have the throughput required to be an efficient firewall. Texas A&M University (TAMU) is in the process of developing another version of their PC packet filtering firewall which could possibly have more efficient source code and thus increase the throughput. Possibly, a Unix workstation may be required to gain the performance needed for the firewall. However, use of a workstation may introduce other security issues not inherent with PCs.

The firewall from TAMU is very basic. More robust diagnostics would aid in troubleshooting. The current version of the firewall has no diagnostics.

This research did not include security holes created by the dial-in access to the NPS network, it focused only on Internet access. Further research should be performed to determine the best way to firewall the dial in access in addition to Internet access.

The firewall used in this research was not formally validated. It is essential that the firewall be proven to be secure itself, using formal methods, before it is used to secure the network.

APPENDIX A: NPS NETWORK SECURITY POLICY

GENERAL NPS COMPUTING AND SECURITY PHILOSOPHY

NPS is primarily an academic institution. The "customers" of NPS computer systems are the students, faculty and staff who are carrying out our mission of education and research. Wherever possible, they should be able to accomplish their tasks as easily as possible, with minimal risk of loss, damage or interference with their work. The primary function of computer systems policies and systems administrators should be to support these goals. NPS encourages the widest possible use of our computing facilities by faculty, staff, students, and other appropriate individuals, subject to the following qualifications:

1. Fraud, illegal acts, willful damage, or violations of laws or appropriate regulations will not be tolerated.

2. No individual may use NPS computing facilities in any manner that significantly impairs or threatens to impair the ability of other legitimate individuals to equally use those same facilities.

3. Individuals who do not use NPS computing facilities in a responsible manner may be denied further access to those facilities.

4. All NPS computer systems (including both local and wide area networks) are intended to be used FOR OFFICIAL USE ONLY. While NPS will interpret "official use" in the broadest sense, i.e. to encompass all activity related to instruction or research, the use of NPS systems for purposes of personal or private gain is strictly prohibited.

Users or systems administrators who feel that specific aspects of these policies unduly interfere with support of their instruction or research programs may request that the Command ADP Security Officer (ADPSO) waive that policy. Such requests will be considered on a case-by-case basis. Users of systems administrators who feel specific policies are inadequate or insufficient should also confer with the ADPSO. In deciding whether to grant requested waivers, or modify aspects of campus security policy, the ADPSO should solicit the advice of senior campus systems administrators.

NPS NETWORK SECURITY POLICY

1. User Identification

a. Every individual NPS user of networked computer systems will have a single, unique campus-wide user name, UNIX numeric identifier, and mail alias. The Computer Center will maintain the official central registry of all assigned user names, and make this information readily available to all campus systems administrators.

b. No individual should be granted a regular account on an NPS system until they have received a formal ADP security briefing, been provided with a written list of NPS mandatory security practices and signed an NPS ADP User Security Policy Agreement. The ADPSO will develop and provide the necessary supporting materials.

c. No individuals other than NPS faculty, staff and students should be granted a regular account on an NPS system without written, recorded and periodically reviewed certification that such access constitutes an appropriate use of federal, DOD or DON resources. The Dean of Computer and Information services will establish criteria for granting such access.

d. The Computer Center will establish and maintain the necessary data bases and gateways to permit intra-campus delivery of e-mail based only on a user's mail alias, and delivery of SMTP- compliant Internet e-mail based only the address:

mailalias@nps.navy.mil

2. System Access

a. All NPS systems should require users to provide positive identification (normally by providing a valid NPS user name and password) prior to being permitted remote login to any networked NPS systems. User passwords should not be able to be determined ("cracked") by any commonly available, public domain programs, and, if possible, should be stored in files which are "hidden" from general users.

b. "Trusted" access to networked NPS systems, i.e. granting login-equivalent access from a remote machine without requiring password authentication, based on the requesting user's presumed positive identification on the remote system, should be permitted only

when the security of the remote machine can be reasonably assured. System administrators should individually approve all trusted remote access to their systems.

c. Root (superuser/supervisor/system administrator) access to networked, multiuser NPS systems should be restricted as much as practical, consistent with mission requirements, and, should be limited to individuals with an appropriate formal computer security training. The ADPSO should develop and coordinate all such training.

d. Shared accounts (where more than a single individual knows the password) should be discouraged wherever possible. When no other feasible method will work, such accounts should be strictly controlled, limited to only the minimal functionality necessary to accomplish their purpose, and located on isolated systems wherever possible.

e. All systems will display, upon user login, a banner clearly identifying appropriate system usage restrictions.

f. Wherever possible, the Computer Center should operate campus-wide versions of those services which provide the greatest risk of security "holes," and especially of any service that will accept anonymous logins. Departments should generally not establish similar services at the departmental level.

g. Network access to NPS or to any NPS system may be denied when the request originates from another system (either on or off-campus) which does not meet minimal security requirements, or which has previously attempted to violate NPS security policies.

h. All dial-in modems connections to any NPS system should be registered with and approved by the ADPSO.

3. File Permissions/Data Security

a. No user other than a file's owner should be able to read or modify that file except as the result of a prior conscious, positive action by the owner. Each individual user should be held responsible for ensuring the propriety of allowing others to read or access any of that user's files.

b. Wherever possible, files intended to be regularly shared by groups of individuals should be placed in separate, group directories, not the personal directory of the originating user.

c. Individual users should be permitted to encrypt their files, if they wish. The Computer Center will provide and support a standard file encryption package.

4. Data Integrity

a. Each multiuser, networked system should establish and provide to all users a written file backup policy.

b. Each multiuser system that does not backup stored user files at least weekly should provide simple means by which individuals may back up their own files.

c. Centrally administered, multiuser systems should not delete user data files without first either obtaining prior positive permission or making an archival copy.

5. Security Monitoring and Reporting

a. All centrally managed, multiuser systems will record (log) and report appropriate security-related events. The ADPSO will provide, at least quarterly, a minimal list of such events.

b. All centrally managed, multiuser systems will regularly check and immediately report any evidence of tampering, etc., with key system and application programs. (Tampering, in this context, includes all computer viruses). The ADPSO will provide, at least quarterly, a minimal list of files to be checked. The Computer Center will provide and support the necessary software.

c. The Computer Center should establish gateway systems which are be capable of monitoring all incoming and outgoing network communications, and of being programmed to recognize and report on specific, software-selectable events

6. Ongoing Vulnerability Assessment and Countermeasures

a. The ADPSO will establish and coordinate, with the advice of senior campus systems administrators, an ongoing program to evaluate potential new security vulnerabilities,

assess the risk to NPS of each, evaluate potential countermeasures, and determine the relative criticality of implementing any specific countermeasure.

b. The ADPSO, with the advice of senior campus systems administrators, will establish and operate an ongoing network security testing program. No individual, however, other than the ADPSO or his designated representative will actively test the security of any system without the prior agreement of the responsible system administrator. Individual users should not possess security testing or defeating software unless they have also specifically notified the administrator of the system where that software is stored and/ or run, and can establish a valid purpose for having such software.

7. Classified Information

Classified information will not be stored or processed on unaccredited NPS systems.

8. Safeguarding Network Operability

a. The NPS backbone should be under the control of the NPS Computer Center. Physical access to the backbone should be limited to Computer Center-authorized personnel.

b. All simultaneous connections to both the NPS backbone network (or any of its subnets) and any other off-campus network(s) must be approved in advance.

c. All network segments, connections, and subnetworks should be designed for maximum redundancy, modularity, and fault tolerance.

9. Responsibilities for ADP Security

a. The Performance Plans for Department Heads and computer systems administration staff should include a critical element addressing their system's compliance with NPS ADP Security Policies.

b. Any computer user who willfully or consistently violates ADP Security policies may have appropriate action taken against them, including possible loss of user privileges on all NPS systems.

APPENDIX B: COMMON IP PROTOCOLS

Table 8 and Table 9 contain a list of the common IP protocols supported by both TCP and UDP. Table 8 is the list of protocols running on the privileged ports, those ports below 1024, and Table 9 is the list of protocols running on ports equal to or above port 1024. The tables are a consolidation of protocols listed in the following places:

- [CHES 94]
- /etc/services file on the SunOS 4.1.3.system
- /etc/services file on the Solaris 2.3 system
- /etc/services file on the Silicon Graphics IRIX 4.0.5 system
- /etc/services file on the Silicon Graphics IRIX 5.2 system
- /etc/services file on the Hewlett Packard HP-UX 9.0.1 system
- /etc/services file on the Digital Equipment Corporation Ultrix 4.3 system
- /etc/services file on the Cray UNICOS 7.0 system

Port	Common IP Protocol	Type of Service	Description
1	tcpmux	TCP	The TCP port multiplexer.
7	echo	TCP / UDP	The echo server. Useful for seeing if a machine is alive.
9	discard	TCP / UDP	The /dev/null of the Internet.
11	systat	TCP	Occasionally connected to netstat, w, or ps.
13	daytime	TCP / UDP	The time of day, in human-readable form.
15	netstat	TCP	This should not be done outside of local network.
17	qotd	TCP	Quote of the day.
19	chargen	TCP / UDP	A character stream generator.
20	ftp-data	TCP	Data channel for file transfer protocol.
21	ftp	TCP	Control channel for file transfer protocol.
23	telnet	TCP	Telecommunications network protocol.
25	smtp	TCP	Simple mail transfer protocol.
37	time	TCP / UDP	Time of day, in machine-readable form.
39	rlp	UDP	Resource location.
42	name	TCP	IEN 116.
43	whois	TCP	Who is user name direcotry server.

Table 8: IP Protocols On Privileged Ports, Less Than 1024

Port	Common IP Protocol	Type of Service	Description
53	domain	TCP / UDP	Domain name service.
57	mtp	TCP	Deprecated
67	bootp	UDP	Bootp server.
68	bootpc	UDP	Bootp client.
69	tftp	UDP	Trivial file transfer protocol.
70	gopher	TCP	Gopher documentation server.
77	rje	TCP	Remote job entry over the network.
79	finger	TCP	Display information about users.
83	http	TCP	Hypertext transfer protocol, sometimes known as world wide web (WWW).
87	link	TCP	
88	kerberos	UDP	Official Kerberos port.
95	supdup	TCP	
101	hostnames	TCP	Usually from sri-nic (network information center).
102	iso-tsap	TCP	
103	x400	TCP	International Standards Organization (ISO) mail server.
104	x400-snd	TCP	ISO mail client.
105	csnet-ns	TCP	
109	pop-2	TCP	Post office protocol version 2.
110	pop-3	TCP	Post office protocol version 3.
111	sunrpc	TCP / UDP	Sun remote procedure call, portmap. NPS and NIS use this service.
113	auth	TCP	Authentication service.
115	sftp	TCP	Simple file transfer protocol.
117	uucp-path	TCP	Unix to Unix copy path service.
119	nntp	TCP	Network news transport protocol.
121	erpc	UDP	SGI, IRIX 5.2.
123	ntp	UDP	Network time protocol.
135	loc-srv	TCP / UDP	NCS local location broker.
137	netbios_ns	TCP / UDP	NetBIOS name service.
138	netbios_dgm	TCP / UDP	NetBIOS datagram service.

Table 8: IP Protocols On Privileged Ports, Less Than 1024

Port	Common IP Protocol	Type of Service	Description
139	netbios_ssn	TCP / UDP	NetBIOS session service.
143	imap2	TCP	SGI, IRIX 5.2.
144	NeWS	TCP	Network extensible window system for Sun.
152	bftp	TCP	Background file transfer protocol.
153	sgmp-netview	TCP	Cray.
153	sgmp	UDP	Cray.
160	sgmp-trap	TCP	Cray.
161	snmp	UDP	Simple network management protocol.
162	snmp-trap	UDP / UDP	Cray.
167	snmp-rt	UDP	Simple network management protocol daemon - routed/gated.
170	print-srv	TCP	Network PostScript.
175	vmnet	TCP	Cray.
177	xdmcp	UDP	X display manager control protocol.
260	rtape	TCP	Remote tape device.
371	albd	UDP	Location broker.
512	exec	TCP	rexec - Remote execution of commands.
512	biff	UDP	Notification of incoming mail messages.
513	login	TCP	rlogin - Remote logins possibly without passwords.
513	who	UDP	Who is logged in.
514	shell	TCP	rsh and rcp - Remote shells without passwords.
514	syslog	UDP	System logs. If this is open, system logs can be attacked.
515	printer	TCP	Remote printer support. It is rarely a good reason for outsiders to use the NPS printers.
517	talk	UDP	Talk to another user.
518	ntalk	UDP	New talk, conversation.
520	efs	TCP	Ultrix.
520	route	UDP	Network routing.
525	timed	UDP	Time server.
526	tempo	TCP	New date.
530	courier	TCP	Experimental.

Table 8: IP Protocols On Privileged Ports, Less Than 1024

Port	Common IP Protocol	Type of Service	Description
531	conference	TCP	
532	netnews	TCP	
533	netwall	UDP	Emergency network broadcasts.
540	uucp	TCP	Unix to Unix copy daemon
543	klogin	TCP	Kerberos "rlogin".
544	kshell	TCP	Kerberos remote shell.
550	new-rwho	UDP	Experimental.
556	remotefs	TCP	Brunhoff remote filesystem.
560	rmonitor	UDP	Experimental.
561	monitor	UDP	Experimental.
570	eval	TCP	Ultrix
600	pcserver	TCP	experimental
601	ta-rauth	TCP	AFS remote authentication.
607	nqs	TCP	Cray network queueing system (NQS).
608	cde	TCP	Cray distributed editor.
609	nqsdev	TCP	Cray NQS development.
610	nqstst	TCP	Cray NQS test system.
704	elcsd	UDP	errlog
705	auditd	TCP	Audit daemon.
712	vexec	TC P	Cray.
713	vlogin	TCP	Cray.
714	vshell	TCP	Cray.
750	kerberos	TCP / UDP	Original kerberos server.
751	hesupd	TCP	Hesiod update daemon
752	krb_prop	TCP	Kerberos propagation.
753	userreg_server	UDP	Cray.
755	kadmind	TCP	Cray.
760	krbupdate	TCP	Kerberos registration.
761	kpasswd	TCP	Kerberos "passwd".
888	erlogin	TCP	Cray.
906	sample	TCP	Cray.
987	DAServer	TCP	SQL distributed access, HP-UX.

Table 8: IP Protocols On Privileged Ports, Less Than 1024

Port	Common IP Protocol	Type of Service	Description
999	applix	UDP	Ultrix.

Table 8: IP Protocols On Privileged Ports, Less Than 1024

Port	Common IP Protocol	Type of Service	Description
1025	listener	TCP	System V Release 3 listener.
1109	kpop	TCP	Cray.
1234	mkuserd	TCP	Cray.
1260	rlb	TCP	Remote loopback diagnostic, HP-UX.
1400	usim	TCP / UDP	usim user interface.
1524	ingreslock	TCP	Ingres database lock.
1525	orasrv	TCP	Oracle database server, Cray.
1536	nft	TCP	NS network file transfer, HP-UX.
1571	crj1	TCP	Cray Japan.
1572	crj2	TCP	Cray Japan.
2000	openwin	TCP	Openwindows window manager, Sun.
2001	lgc_pd	TCP	Cray.
2002	lgc_sdsrv	TCP	Cray.
2003	craydiag	TCP / UDP	Cray online diagnostics.
2015	cypress	TCP	Cray.
2017	cypress-stat	TCP	Cray.
2030	client	TCP	Cray.
2049	nfs	UDP	Network file system, Cray.
2053	knetd	TCP	Cray.
2105	eklogin	TCP	Kerberos encrypted "rlogin".
2106	venus.itc	TCP	Cray.
2049	nfsd	UDP	Network filesystem daemon.
2766	listen	TCP	System V listener. Like tcpmux, but with more services.
4045	lockd	TCP / UDP	Network filesystem lock daemon and manager.
4672	rfa	TCP	NS remote file access, HP-UX.

Table 9: IP Protocols Equal To Or Above Port 1024

Port	Common IP Protocol	Type of Service	Description
5000	VTVIN	TCP	Cray.
5003	xcwcs	TCP	Cray.
5004	cwdata	TCP	Cray.
5130	sgi-dogfight	TCP	SGI.
5131	sgi-arena	UDP	SGI.
5133	sgi-bznet	UDP	SGI BZ demo port.
5135	sgi-objectserver	UDP	SGI object server.
5136	sgi-directoryserver	UDP	SGI directory server.
5137	sgi-oortnet	UDP	Oort port, SGI.
5232	sgi-dgl	TCP	SGI distributed graphics library.
5555	sbm-comm	UDP	Space boulders port, SGI.
5558	remotefb	TCP	SGI distributed graphics, Cray.
5696	langmgrx.osB	TCP	LAN Manager/X for B.00.00 OfficeShare, HP-UX.
5710	hcserver	TCP	HP cooperative services, HP-UX.
5999	grmd	TCP	Graphics resource manager, HP-UX.
	MBone	UDP	
6000 thru 6xxx	X11 Window System	TCP	MIT's windowing system
6667	IRC	TCP	Internet relay chat.
7489	iasqlsrv	TCP	Information access, HP-UX.

Table 9: IP Protocols Equal To Or Above Port 1024

APPENDIX C: CURRENT NETWORK TRAFFIC DATA RESULTS

INTERNET BYTES TRANSMITTED OVER ONE WEEK

The raw data results of monitoring over a one week period are given in Table 10. The data was recorded once at the beginning of the business day, between 8:00 and 9:00 am, and once at the end of the business day, between 5:00 and 6:00 pm.

Start Day and Time	Stop Day and Time	BARRNET Number of Bytes Sent	BARRNET Number of Bytes Received	DDN Number of Bytes Sent	DDN Number of Bytes Received
Friday 17:00:19	Saturday 07:50:03	881,351,666	140,161,325	104,142,214	10,172,750
Saturday 07:52:59	Saturday 16:56:55	870,894,426	419,726,591	63,863,624	5,549,940
Saturday 16:59:04	Sunday 07:57:55	1,090,000,000	439,248,791	82,695,231	4,124,008
Sunday 08:00:08	Sunday 18:10:20	665,443,363	173,492,576	43,509,633	5,924,565
Sunday 18:12:46	Monday 08:01:30	751,690,273	195,556,572	58,874,635	2,224,247
Monday 08:03:27	Monday 16:34:05	712,618,300	191,442,367	36,348,748	1,437,948
Monday 16:35:27	Tuesday 07:55:59	1,060,000,000	410,950,569	115,091,959	9,197,780
Tuesday 07:57:45	Tuesday 16:56:51	762,152,646	292,358,401	73,192,497	13,428,544
Tuesday 16:58:26	Wednesday 07:53:25	522,760,437	352,363,189	94,243,384	7,123,194
Wednesday 07:54:42	Wednesday 17:02:43	629,102,333	799,762,034	62,006,206	9,221,776
Wednesday 17:04:08	Thursday 08:55:52	1,914,423,858	422,819,595	92,782,173	6,665,960
Thursday 08:57:44	Thursday 17:08:52	773,793,515	818,399,363	12,084,254	2,006,776

Table 10: Internet Bytes Transferred for One Week

Start Day and Time	Stop Day and Time	BARRNET Number of Bytes Sent	BARRNET Number of Bytes Received	DDN Number of Bytes Sent	DDN Number of Bytes Received
Thursday 17:12:31	Friday 08:16:41	3,390,000,000	703,486,816	912,647	741,009

Table 10: Internet Bytes Transferred for One Week

INTERNET BYTES TRANSMITTED OVER ONE DAY

The raw results of monitoring over a one day period, Wednesday, are given in Table 11. The data was recorded approximately every 15 minutes between 8:00 am until 11:00 pm.

Start Time Wednesday	Stop Time Wednesday	BARRNET Number of Bytes Sent	BARRNET Number of Bytes Received	DDN Number of Bytes Sent	DDN Number of Bytes Received
07:54:42	08:14:13	20,258,025	5,967,368	1,840,811	472,611
08:15:37	08:29:08	15,153,285	12,895,282	1,591,060	168,622
08:30:10	08:45:14	23,093,269	7,678,223	1,505,894	247,626
08:46:40	09:01:23	33,467,818	6,314,431	1,539,344	226,927
09:02:56	09:17:25	7,969,511	15,280,687	1,542,060	206,407
09:18:33	09:32:37	5,858,737	12,025,965	2,074,479	327,374
09:34:01	09:48:42	18,941,855	8,290,514	2,565,139	594,329
09:49:54	10:05:00	15,384,330	12,291,614	1,249,043	251,767
10:06:04	10:19:26	25,458,480	25,822,456	1,145,271	225,340
10:20:45	10:35:29	16,818,652	28,161,769	1,188,215	237,350
10:36:28	10:51:09	15,793,395	22,442,905	1,408,616	254,133
10:52:21	11:05:28	44,634,957	22,919,551	2,455,798	302,616
11:06:39	11:18:55	40,493,795	19,011,797	1,487,610	227,625
11:20:08	11:34:05	9,269,983	48,894,702	1,652,523	193,632
11:36:18	11:50:49	8,650,691	16,462,187	1,297,290	159,820
11:53:01	12:06:08	5,530,835	12,740,205	1,750,866	210,525
12:07:59	12:22:45	7,426,190	15,966,405	2,883,888	305,695
12:24:23	12:39:21	15,007,978	28,274,270	4,419,330	480,848

Table 11: Internet Bytes Transferred for One Day

Start Time Wednesday	Stop Time Wednesday	BARRNET Number of Bytes Sent	BARRNET Number of Bytes Received	DDN Number of Bytes Sent	DDN Number of Bytes Received
12:40:39	12:55:11	12,391,936	26,381,887	2,738,146	226,358
12:56:28	13:11:25	16,203,261	42,886,222	2,210,410	266,747
13:13:14	13:28:01	8,929,681	60,299,968	2,295,041	688,704
13:30:19	13:44:37	7,631,890	56,310,886	1,615,119	183,956
13:46:01	14:00:44	12,685,027	49,312,160	2,202,584	274,347
14:01:44	14:16:26	10,642,178	41,153,482	1,658,951	158,883
14:18:08	14:41:44	24,986,420	57,155,779	3,166,027	821,140
14:43:33	14:58:52	9,691,042	28,351,392	1,625,656	236,345
15:00:44	15:15:46	8,038,496	21,911,503	1,385,800	224,692
15:17:00	15:31:38	10,036,248	17,244,627	1,157,911	171,366
15:32:39	15:47:49	13,367,166	13,474,885	1,352,683	97,335
15:49:38	16:13:15	58,407,962	22,283,848	2,413,763	284,411
16:14:18	16:29:01	34,542,320	12,949,092	1,856,114	193,942
16:31:51	16:46:26	31,961,562	12,732,657	1,490,776	177,730
16:48:07	17:02:48	40,375,358	15,873,315	1,239,988	122,573
17:04:08	17:20:05	30,084,576	23,395,228	1,465,847	140,372
17:21:29	17:41:13	27,944,974	4,560,814	1,797,341	125,468
17:42:21	17:57:01	12,711,735	3,581,809	1,399,654	82,257
17:58:43	18:14:11	11,669,131	5,350,140	1,561,579	89,808
18:15:50	18:32:00	13,556,349	4,495,981	1,770,023	121,672
18:34:53	18:47:57	9,236,994	3,337,698	1,097,189	91,717
18:49:42	19:16:08	22,933,207	7,426,891	2,314,541	155,703
19:17:25	19:31:28	12,075,427	4,099,931	1,183,133	75,921
19:32:11	19:47:29	14,901,752	3,153,632	1,308,145	68,313
19:48:50	20:04:42	10,509,078	4,630,377	1,448,232	85,392
20:06:29	20:24:14	20,684,384	4,289,721	1,671,933	94,946
20:25:40	20:40:28	22,736,549	3,449,373	1,221,135	69,513
20:41:06	21:11:40	145,956,992	12,317,180	2,523,575	146,746
21:13:53	21:32:15	121,509,993	9,818,531	2,529,688	217,826
21:33:38	21:49:43	96,981,054	8,584,547	2,037,621	158,194
21:50:56	22:06:41	105,333,532	7,203,651	1,642,007	129,962

Table 11: Internet Bytes Transferred for One Day

Start Time Wednesday	Stop Time Wednesday	BARRNET Number of Bytes Sent	BARRNET Number of Bytes Received	DDN Number of Bytes Sent	DDN Number of Bytes Received
22:09:07	22:26:35	114,502,538	7,693,459	2,057,342	163,544
22:35:05	22:51:51	101,095,593	6,495,085	3,527,892	321,644

Table 11: Internet Bytes Transferred for One Day

INTERNET DATA RATES OVER ONE WEEK

The calculated results from the raw data (Table 10 on page 75) is presented below in Table 12. The total elapsed time is the total number of elapsed seconds from the start and stop time listed in Table 10 on page 75. The total number of bytes transferred are the total BARRNET and DDN bytes also listed in Table 10. The average number of kilobits per second transferred to and from the BARRNET and DDN routers, including frame overhead, was calculated to be 306.81 kilobits per second for the one week monitoring period.

Total Elapsed Time seconds	Total Number of Bytes Transferred	Total Number of Bits Transferred	Total Number of Bits/Second Transferred	Total Number of Bits/Second Transferred Including Frame Overhead	Total Number of Kbits/sec Trans-ferred
53,384	1,135,827,955	9,086,623,640	170,212.49	180,900.13	180.90
32,636	1,360,034,581	10,880,276,648	333,382.66	354,315.76	354.32
53,931	1,616,068,030	12,928,544,240	239,723.80	254,776.06	254.78
36,612	888,370,137	7,106,961,096	194,115.62	206,304.14	206.30
49,724	1,008,345,727	8,066,765,816	162,230.83	172,417.30	172.42
30,638	941,847,363	7,534,778,904	245,929.20	261,371.10	261.37
55,232	1,595,240,308	12,761,922,464	231,060.30	245,568.58	245.57
32,346	1,141,132,088	9,129,056,704	282,231.40	299,952.70	299.95
53,699	976,490,204	7,811,921,632	145,476.11	154,610.55	154.61
32,881	1,500,092,349	12,000,738,792	364,974.87	387,891.65	387.89

Table 12: Internet Data Rates for One Week

Total Elapsed Time seconds	Total Number of Bytes Transferred	Total Number of Bits Transferred	Total Number of Bits/Second Transferred	Total Number of Bits/Second Transferred Including Frame Overhead	Total Number of Kbits/sec Trans-ferred
57,104	2,436,691,586	19,493,532,688	341,368.95	362,803.51	362.80
29,468	1,606,283,908	12,850,271,264	436,075.45	463,456.62	463.46
54,250	4,095,140,472	32,761,123,776	603,891.68	641,810.04	641.81

Table 12: Internet Data Rates for One Week

INTERNET DATA RATES OVER ONE DAY

The calculated results from the raw data (Table 11 on page 76) is presented below in Table 13. The total elapsed time is the total number of elapsed seconds from the start and stop time listed in Table 11 on page 76. The total number of bytes transferred are the total BARRNET and DDN bytes also listed in Table 11. The average number of Kbits per second transferred to and from the BARRNET and DDN routers, including frame overhead, was calculated to be 438.17 kilobits per second for the one day monitoring period.

Total Elapsed Time seconds	Total Number of Bytes Transferred	Total Number of Bits Transferred	Total Number of Bits/Second Transferred	Total Number of Bits/Second Transferred Including Frame Overhead	Total Number of Kbits/sec Transferred
1,181	28,538,815	228,310,520	193,319.66	205,458.20	205.46
811	29,808,249	238,465,992	294,039.45	312,502.18	312.50
904	32,525,012	260,200,096	287,831.96	305,904.93	305.90
883	41,548,520	332,388,160	376,430.53	400,066.61	400.07
869	24,998,665	199,989,320	230,137.31	244,587.63	244.59
844	20,286,555	162,292,440	192,289.62	204,363.49	204.36
881	30,391,837	243,134,696	275,975.82	293,304.34	293.30

Table 13: Internet Data Rates for One Day

Total Elapsed Time seconds	Total Number of Bytes Transferred	Total Number of Bits Transferred	Total Number of Bits/Second Transferred	Total Number of Bits/Second Transferred Including Frame Overhead	Total Number of Kbits/sec Transferred
906	29,176,754	233,414,032	257,631.38	273,808.06	273.81
802	52,651,547	421,212,376	525,202.46	558,179.93	558.18
884	46,405,986	371,247,888	419,963.67	446,333.19	446.33
881	39,899,049	319,192,392	362,306.91	385,056.17	385.06
787	70,312,922	562,503,376	714,743.81	759,622.57	759.62
736	61,220,827	489,766,616	665,443.77	707,226.99	707.23
837	60,010,840	480,086,720	573,580.31	609,595.42	609.60
871	26,569,988	212,559,904	244,041.22	259,364.57	259.36
787	20,232,431	161,859,448	205,666.39	218,580.18	218.58
886	26,582,178	212,657,424	240,019.67	255,090.50	255.09
898	48,182,426	385,459,408	429,242.10	456,194.21	456.19
872	41,738,327	333,906,616	382,920.43	406,964.01	406.96
897	61,566,640	492,533,120	549,089.32	583,566.64	583.57
887	72,213,394	577,707,152	651,304.57	692,199.98	692.20
858	65,741,851	525,934,808	612,977.63	651,466.50	651.47
883	64,474,118	515,792,944	584,136.97	620,814.93	620.81
882	53,613,494	428,907,952	486,290.20	516,824.36	516.82
1,416	86,129,366	689,034,928	486,606.59	517,160.62	517.16
919	39,904,435	319,235,480	347,372.67	369,184.20	369.18
902	31,560,491	252,483,928	279,915.66	297,491.57	297.49
878	28,610,152	228,881,216	260,684.76	277,053.15	277.05
910	28,292,069	226,336,552	248,721.49	264,338.71	264.34
1,417	83,389,984	667,119,872	470,797.37	500,358.74	500.36
883	49,541,468	396,331,744	448,846.82	477,029.91	477.03
875	46,362,725	370,901,800	423,887.77	450,503.68	450.50
881	57,611,234	460,889,872	523,144.01	555,992.22	555.99
957	55,086,023	440,688,184	460,489.22	489,403.34	489.40
1,184	34,428,597	275,428,776	232,625.66	247,232.22	247.23
880	17,775,455	142,203,640	161,595.05	171,741.60	171.74

Table 13: Internet Data Rates for One Day

Total Elapsed Time seconds	Total Number of Bytes Transferred	Total Number of Bits Transferred	Total Number of Bits/Second Transferred	Total Number of Bits/Second Transferred Including Frame Overhead	Total Number of Kbits/sec Transferred
928	18,670,658	149,365,264	160,953.95	171,060.25	171.06
970	19,944,025	159,552,200	164,486.80	174,814.93	174.81
784	13,763,598	110,108,784	140,444.88	149,263.41	149.26
1,586	32,830,342	262,642,736	165,600.72	175,998.79	176.00
843	17,434,412	139,475,296	165,451.12	175,839.80	175.84
918	19,431,842	155,454,736	169,340.67	179,973.57	179.97
952	16,673,079	133,384,632	140,109.91	148,907.41	148.91
1,065	26,740,984	213,927,872	200,871.24	213,483.95	213.48
888	27,476,570	219,812,560	247,536.67	263,079.49	263.08
1,834	160,944,493	1,287,555,944	702,047.95	746,129.54	746.13
1,102	134,076,038	1,072,608,304	973,328.77	1,034,444.08	1,034.44
965	107,761,416	862,091,328	893,358.89	949,452.89	949.45
945	114,309,152	914,473,216	967,696.52	1,028,458.19	1,028.46
1,048	124,416,883	995,335,064	949,747.20	1,009,381.83	1,009.38
1,006	111,440,214	891,521,712	886,204.49	941,849.26	941.85

Table 13: Internet Data Rates for One Day

APPENDIX D: DOIT.SH SCRIPT, TTEST.SH SCRIPT, AND NTTCP PROGRAM

```
#!/bin/sh

date > start
date > run1_start_time

ttest.sh 65536 512
ttest.sh 8192 512
ttest.sh 65536 1024
ttest.sh 8192 1024
ttest.sh 65536 2048
ttest.sh 8192 2048
ttest.sh 65536 4096
ttest.sh 8192 4096

date > run1_finish_time

mkdir run1
mv *.log *.out run1/.
mv *time run1/.

date > run2_start_time

ttest.sh 65536 512
ttest.sh 8192 512
ttest.sh 65536 1024
ttest.sh 8192 1024
ttest.sh 65536 2048
ttest.sh 8192 2048
ttest.sh 65536 4096
ttest.sh 8192 4096

date > run2_finish_time

mkdir run2
mv *.log *.out run2/.
mv *time run2/.

date > run3_start_time

ttest.sh 65536 512
ttest.sh 8192 512
ttest.sh 65536 1024
ttest.sh 8192 1024
ttest.sh 65536 2048
 ttest.sh 8192 2048
 ttest.sh 65536 4096
 ttest.sh 8192 4096

 date > run3_finish_time
 mkdir run3
 mv *.log *.out run3/.
 mv *time run3/.

 date > run4_start_time

 ttest.sh 65536 512
 ttest.sh 8192 512
 ttest.sh 65536 1024
 ttest.sh 8192 1024
 ttest.sh 65536 2048
 ttest.sh 8192 2048
 ttest.sh 65536 4096
 ttest.sh 8192 4096

 date > run4_finish_time

 mkdir run4
 mv *.log *.out run4/.
 mv *time run4/.

 date > run5_start_time

 ttest.sh 65536 512
 ttest.sh 8192 512
 ttest.sh 65536 1024
 ttest.sh 8192 1024
 ttest.sh 65536 2048
 ttest.sh 8192 2048
 ttest.sh 65536 4096
 ttest.sh 8192 4096

date > run5_finish_time

mkdir run5
mv *.log *.out run5/.
mv *time run5/.
```

```
date > run6_start_time

ttest.sh 65536 512
ttest.sh 8192 512
ttest.sh 65536 1024
ttest.sh 8192 1024
ttest.sh 65536 2048
ttest.sh 8192 2048
ttest.sh 65536 4096
ttest.sh 8192 4096

date > run6_finish_time

mkdir run6
mv *.log *.out run6/.
mv *time run6/.

date > finish
```

TTEST.SH Script

```
#!/bin/sh
#
# Use nttcp to test network throughput.
# Usage: ttest.sh byte_per_write
                  number_of_writes
#
DATALEN=$1
NPKTS=$2

#White to Gold
RECHOST=131.120.1.2
RSH=/usr/ucb/rsh
NTTCP=nttcp

rm -f ttest.out
rm -f ttest.tran.log
rm -f ttest.recv.log

# from 4KB to 60KB windows in steps of 8KB
SIZE=4
while test $SIZE -lt 61
do
        $RSH $RECHOST $NTTCP -r -w$SIZE
                > tmp1 2>&1 &
        sleep 5
        $NTTCP -t -l$DATALEN -n$NPKTS -w$SIZE
              $RECHOST >> ttest.tran.log 2>&1
        sleep 5
        grep 'Mb/s' tmp1 | awk '{print
'$SIZE'*1024,$12}' >> ttest.out
        cat tmp1 >> ttest.recv.log
      SIZE=`expr $SIZE + 8`
done

rm -f tmp1
mv ttest.out ttest.$DATALEN.$NPKTS.out
mv ttest.tran.log
      ttest.$DATALEN.$NPKTS.tran.log
mv ttest.recv.log
       ttest.$DATALEN.$NPKTS.recv.log
```

NTTCP Program

```
/*
 *      N T T C P . C
 *
 * Test TCP connection.  Makes a connection on port 2000
 * and transfers zero buffers or data copied from stdin.
 *
 * Usable on 4.2, 4.3, and 4.1a systems by defining one of
 * BSD42 BSD43 (BSD41a)
 *
 * Modified for operation under 4.2BSD, 18 Dec 84
 *     T.C. Slattery, USNA
 * Minor improvements, Mike Muuss and Terry Slattery, 16-Oct-85.
 *
 * Modified on 5 Apr 94 for opertion under Solaris 2.3 based on changes
 * for the TTCP.C program provided by Don Merritt of ARL.
 *     CPT Mark Schivley, USA
 */
#ifndef lint
static char RCSid[] = "@(#)$Header: /src/opt/brl/sbin/ttcp/RCS/ttcp.c,v 1.2 1993/11/30 20:15:39
root Exp $ (BRL)";
#endif
#define BSD43
/* #define BSD42 */
/* #define BSD41a */
#include <stdio.h>
#include <ctype.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <sys/time.h>               /* struct timeval */
#ifdef SYSV
#include <sys/times.h>
#include <sys/param.h>
#else
#include <sys/resource.h>
#endif
#ifdef SYSV
#define bcopy(s,d,l) memcpy(d, s, (size_t) l)
#define bzero(s,l) memset(s, 0, (size_t) l)
#endif
struct sockaddr_in sinme;
struct sockaddr_in sinhim;
struct sockaddr_in sindum;
struct sockaddr_in frominet;
int domain, fromlen;
```

```
int fd;                         /* fd of network socket */
int sendwin = 32 * 1024;
int rcvwin = 32 * 1024;
int optlen = sizeof(int);
int buflen = 1024;              /* length of buffer */
char *buf;                      /* ptr to dynamic buffer */
int nbuf = 1024;                /* number of buffers to send in sinkmode */
int udp = 0;                    /* 0 = tcp, !0 = udp */
int options = 0;                /* socket options */
int one = 1;                    /* for 4.3 BSD style setsockopt() */
short port = 2001;              /* TCP port number */
char *host;                     /* ptr to name of host */
int trans;                      /* 0=receive, !0=transmit mode */
int sinkmode = 1;               /* 0=normal I/O, !0=sink/source mode */
int verbose = 0;
int nodelay = 0;                /* set TCP_NODELAY socket option */
int window = 0;                 /* 0=use default   1=set to specified size*/
struct hostent *addr;
extern int errno;
char Usage[] = "\
Usage: ttcp -t [-options] host <in\n\
        -l##    length of bufs written to network (default 1024)\n\
        -s      don't source a pattern to network, use stdin\n\
        -n##    number of bufs written to network (-s only, default 1024)\n\
        -p##    port number to send to (default 2000)\n\
        -u      use UDP instead of TCP\n\
Usage: ttcp -r [-options] >out\n\
        -l##    length of network read buf (default 1024)\n\
        -s      sink (discard) all data from network\n\
        -p##    port number to listen at (default 2000)\n\
        -B      Only output full blocks, as specified in -l## (for TAR)\n\
        -u      use UDP instead of TCP\n\
";
char stats[128];
double t;                       /* transmission time */
long nbytes;                    /* bytes on net */
int b_flag = 0;                 /* use mread() */
void prep_timer();
double read_timer();
double cput, realt;             /* user, real time (seconds) */
main(argc,argv)
int argc;
char **argv;
{
   unsigned long addr_tmp;
   if (argc < 2) goto usage;
   argv++; argc--;
   while( argc>0 && argv[0][0] == '-' ) {
   switch (argv[0][1]) {
   case 'B':
        b_flag = 1;
```

```
            break;
        case 't':
            trans = 1;
            break;
        case 'r':
            trans = 0;
            break;
        case 'd':
            options |= SO_DEBUG;
            break;
        case 'n':
            nbuf = atoi(&argv[0][2]);
            break;
        case 'l':
            buflen = atoi(&argv[0][2]);
            break;
        case 'w':
            window=1;
            sendwin = 1024 * atoi(&argv[0][2]);
            rcvwin = 1024 * atoi(&argv[0][2]);
            break;
        case 's':
            sinkmode = 1;/* source or sink, really */
            break;
        case 'p':
            port = atoi(&argv[0][2]);
            break;
        case 'u':
            udp = 1;
            break;
        default:
            goto usage;
        }
        argv++; argc--;
        }
        if(trans)  {
        /* xmitr */
        if (argc != 1) goto usage;
        bzero((char *)&sinhim, sizeof(sinhim));
        host = argv[0];
        if (atoi(host) > 0 )  {
            /* Numeric */
            sinhim.sin_family = AF_INET;
#ifdef cray
            addr_tmp = inet_addr(host);
            sinhim.sin_addr = addr_tmp;
#else
            sinhim.sin_addr.s_addr = inet_addr(host);
#endif
        } else {
            if ((addr=gethostbyname(host)) == NULL)
```

```
            err("bad hostname");
        sinhim.sin_family = addr->h_addrtype;
        bcopy(addr->h_addr,(char*)&addr_tmp, addr->h_length);
#ifdef cray
        sinhim.sin_addr = addr_tmp;
#else
        sinhim.sin_addr.s_addr = addr_tmp;
#endif cray
    }
    sinhim.sin_port = htons(port);
    sinme.sin_port = 0;/* free choice */
    } else {
    /* rcvr */
    sinme.sin_port =  htons(port);
    }
    if( (buf = (char *)malloc(buflen)) == (char *)NULL)
    err("malloc");
    fprintf(stderr,"ttcp%s: nbuf=%d, buflen=%d, port=%d\n",
    trans?"-t":"-r",
    nbuf, buflen, port);
    if ((fd = socket(AF_INET, udp?SOCK_DGRAM:SOCK_STREAM, 0)) < 0)
    err("socket");
    mes("socket");
/* Try the getsockopt & setsockopt for Solaris here */
#ifndef SOLARIS
    if (bind(fd, &sinme, sizeof(sinme)) < 0)
    err("bind");
#else
    /*
     * Under Solaris, calling connect() on a stream socket binds the
     * socket to an address. If a bind() is done before the connect(),
     * an error "connect: Address family not supported by protocol family"
     * results. Only call bind() for the cases where you're not going
     * to call connect().
     */
    if (udp || (!udp && !trans) )
    if (bind(fd, (struct sockaddr *) &sinme, sizeof(sinme)) < 0)
        err("bind");
#endif /* SOLARIS */
    if (!udp)  {
      if (trans) {
    /* We are the client if transmitting */
    if(options)  {
#ifdef BSD42
        if( setsockopt(fd, SOL_SOCKET, options, 0, 0) < 0)
#else BSD43
#ifndef SOLARIS
        if( setsockopt(fd, SOL_SOCKET, options,  &one, sizeof(one)) < 0)
#else
        if( setsockopt(fd, SOL_SOCKET, options, (char *) &one, sizeof(one)) <
0)
```

```
#endif /* SOLARIS */
#endif
                err("setsockopt");
    }
#ifndef SOLARIS
    if(connect(fd,  &sinhim, sizeof(sinhim) ) < 0) {
#else
    if(connect(fd, (struct sockaddr *) &sinhim, sizeof(sinhim) ) < 0) {
#endif /* SOLARIS */
          err("connect");
    }
    mes("connect");
   if(window){
    if (setsockopt (fd, SOL_SOCKET, SO_SNDBUF, (char *) &sendwin,
    sizeof(sendwin)) < 0 )
      printf("get send window size didn't work\n");
    if (setsockopt (fd, SOL_SOCKET, SO_RCVBUF, (char *) &rcvwin,
    sizeof(rcvwin)) < 0 )
     printf("get rcv window size didn't work\n");
    if (getsockopt (fd, SOL_SOCKET, SO_RCVBUF, (char *) &sendwin, &optlen) < 0 )
      printf("get send window size didn't work\n");
    else printf("send window size = %d\n", sendwin);
    if (getsockopt (fd, SOL_SOCKET, SO_RCVBUF, (char *) &rcvwin, &optlen) < 0 )
      printf("get rcv window size didn't work\n");
    else printf("receive window size = %d\n", rcvwin);
    }
       } else {
    /* otherwise, we are the server and
          * should listen for the connections
          */
#ifndef SOLARIS
    listen(fd,0);  /* allow a queue of 0 */
#else
       /*
        * Under Solaris, specifying a queue length of 0
        * results in a "connection refused".
        */
    listen(fd,1);
#endif /* SOLARIS */
    if(options)  {
#ifdef BSD42
         if( setsockopt(fd, SOL_SOCKET, options, 0, 0) < 0)
#else BSD43
#ifndef SOLARIS
         if( setsockopt(fd, SOL_SOCKET, options,  &one, sizeof(one)) < 0)
#else
         if( setsockopt(fd, SOL_SOCKET, options, (char *) &one, sizeof(one)) <
0)
#endif /* SOLARIS */
#endif
              err("setsockopt");
```

```
    }
    fromlen = sizeof(frominet);
    domain = AF_INET;
#ifndef SOLARIS
    if((fd=accept(fd,  &frominet, &fromlen) ) < 0)
#else
    if((fd=accept(fd, (struct sockaddr *) &frominet, &fromlen) ) < 0)
#endif /* SOLARIS */
        err("accept");
    mes("accept");
    if (window){
    if (setsockopt (fd, SOL_SOCKET, SO_SNDBUF, (char *) &sendwin,
    sizeof(sendwin)) < 0 )
      printf("get send window size didn't work\n");
    if (setsockopt (fd, SOL_SOCKET, SO_RCVBUF, (char *) &rcvwin,
    sizeof(rcvwin)) < 0 )
     printf("get rcv window size didn't work\n");
    if (getsockopt (fd, SOL_SOCKET, SO_RCVBUF, (char *) &sendwin, &optlen) < 0 )
      printf("get send window size didn't work\n");
    else printf("send window size = %d\n", sendwin);
    if (getsockopt (fd, SOL_SOCKET, SO_RCVBUF, (char *) &rcvwin, &optlen) < 0 )
      printf("get rcv window size didn't work\n");
    else printf("receive window size = %d\n", rcvwin);
    }
    }
    }
    prep_timer();
    errno = 0;
    if (sinkmode) {
    register int cnt;
    if (trans)  {
        pattern( buf, buflen );
        if(udp)  (void)Nwrite( fd, buf, 4 ); /* rcvr start */
        while (nbuf-- && Nwrite(fd,buf,buflen) == buflen)
            nbytes += buflen;
        if(udp)  (void)Nwrite( fd, buf, 4 ); /* rcvr end */
    } else {
        while ((cnt=Nread(fd,buf,buflen)) > 0)  {
            static int going = 0;
            if( cnt <= 4 )  {
            if( going )
                break;/* "EOF" */
            going = 1;
            prep_timer();
            } else
            nbytes += cnt;
        }
    }
    } else {
    register int cnt;
    if (trans)  {
```

```
        while((cnt=read(0,buf,buflen)) > 0 &&
          Nwrite(fd,buf,cnt) == cnt)
            nbytes += cnt;
    } else {
        while((cnt=Nread(fd,buf,buflen)) > 0 &&
          write(1,buf,cnt) == cnt)
            nbytes += cnt;
    }
    }
    if(errno) err("IO");
    (void)read_timer(stats,sizeof(stats));
    if(udp&&trans) {
    (void)Nwrite( fd, buf, 4 ); /* rcvr end */
    (void)Nwrite( fd, buf, 4 ); /* rcvr end */
    (void)Nwrite( fd, buf, 4 ); /* rcvr end */
    (void)Nwrite( fd, buf, 4 ); /* rcvr end */
    }
    fprintf(stdout,
     "ttcp%s: %ld bytes in %.2f real seconds = %.2f KB/sec = %.4f Mb/s\n",
          trans?"-t":"-r",
          nbytes, realt, ((double)nbytes)/realt/1024,
                  ((double)nbytes)/realt/128000 );
     if (verbose) {
       fprintf(stdout,
         "ttcp%s: %ld bytes in %.2f CPU seconds = %.2f KB/cpu sec\n",
         trans?"-t":"-r",
         nbytes, cput, ((double)nbytes)/cput/1024 );
    }
    exit(0);
usage:
    fprintf(stderr,Usage);
    exit(1);
}
err(s)
char *s;
{
    fprintf(stderr,"ttcp%s: ", trans?"-t":"-r");
    perror(s);
    fprintf(stderr,"errno=%d\n",errno);
    exit(1);
}
mes(s)
char *s;
{
    fprintf(stderr,"ttcp%s: %s\n", trans?"-t":"-r", s);
}
pattern( cp, cnt )
register char *cp;
register int cnt;
{
    register char c;
```

```
    c = 0;
    while( cnt-- > 0 ) {
    while( !isprint((c&0x7F)) ) c++;
    *cp++ = (c++&0x7F);
    }
}
/******* timing *********/
#ifdef SYSV
extern long time();
#if sgi
static void tvsub();
static structtimeval time0;/* Time at which timeing started */
#else
static long time0;
#endif
static struct tms tms0;
#else
static structtimeval time0;/* Time at which timeing started */
static structrusage ru0;/* Resource utilization at the start */
static void prusage();
static void tvadd();
static void tvsub();
static void psecs();
#endif
/*
 *      PREP_TIMER
 */
void
prep_timer()
{
#ifdef SYSV
#if sgi
    gettimeofday(&time0, (struct timezone *)0);
#else
    (void)time(&time0);
#endif
    (void)times(&tms0);
#else
    gettimeofday(&time0, (struct timezone *)0);
    getrusage(RUSAGE_SELF, &ru0);
#endif
}
/*
 *      READ_TIMER
 *
 */
double
read_timer(str,len)
char *str;
{
#ifdef SYSV
```

```
    long now;
    struct tms tmsnow;
    char line[132];
#ifdef sgi
    struct timeval timedol;
    struct timeval td;
    gettimeofday(&timedol, (struct timezone *)0);
    tvsub( &td, &timedol, &time0 );
    realt = td.tv_sec + ((double)td.tv_usec) / 1000000;
#else
    (void)time(&now);
    realt = now-time0;
#endif
    (void)times(&tmsnow);
    cput = tmsnow.tms_utime - tms0.tms_utime;
    cput /= HZ;
    if( cput < 0.00001 )  cput = 0.01;
    if( realt < 0.00001 )  realt = cput;
    sprintf(line,"%g CPU secs in %g elapsed secs (%g%%)",
    cput, realt,
    cput/realt*100 );
    (void)strncpy( str, line, len );
    return( cput );
#else
    /* BSD */
    struct timeval timedol;
    struct rusage ru1;
    struct timeval td;
    struct timeval tend, tstart;
    char line[132];
    getrusage(RUSAGE_SELF, &ru1);
    gettimeofday(&timedol, (struct timezone *)0);
    prusage(&ru0, &ru1, &timedol, &time0, line);
    (void)strncpy( str, line, len );
    /* Get real time */
    tvsub( &td, &timedol, &time0 );
    realt = td.tv_sec + ((double)td.tv_usec) / 1000000;
    /* Get CPU time (user+sys) */
    tvadd( &tend, &ru1.ru_utime, &ru1.ru_stime );
    tvadd( &tstart, &ru0.ru_utime, &ru0.ru_stime );
    tvsub( &td, &tend, &tstart );
    cput = td.tv_sec + ((double)td.tv_usec) / 1000000;
    if( cput < 0.00001 )  cput = 0.00001;
    return( cput );
#endif
}
#ifndef SYSV
static void
prusage(r0, r1, e, b, outp)
    register struct rusage *r0, *r1;
    struct timeval *e, *b;
```

```
    char *outp;
{
    struct timeval tdiff;
    register time_t t;
    register char *cp;
    register int i;
    int ms;
    t = (r1->ru_utime.tv_sec-r0->ru_utime.tv_sec)*100+
      (r1->ru_utime.tv_usec-r0->ru_utime.tv_usec)/10000+
      (r1->ru_stime.tv_sec-r0->ru_stime.tv_sec)*100+
      (r1->ru_stime.tv_usec-r0->ru_stime.tv_usec)/10000;
    ms =  (e->tv_sec-b->tv_sec)*100 + (e->tv_usec-b->tv_usec)/10000;
#define END(x){while(*x) x++;}
    cp = "%Uuser %Ssys %Ereal %P %Xi+%Dd %Mmaxrss %F+%Rpf %Ccsw";
    for (; *cp; cp++)  {
    if (*cp != '%')
        *outp++ = *cp;
    else if (cp[1]) switch(*++cp) {
    case 'U':
        tvsub(&tdiff, &r1->ru_utime, &r0->ru_utime);
        sprintf(outp,"%d.%01d", tdiff.tv_sec, tdiff.tv_usec/100000);
        END(outp);
        break;
    case 'S':
        tvsub(&tdiff, &r1->ru_stime, &r0->ru_stime);
        sprintf(outp,"%d.%01d", tdiff.tv_sec, tdiff.tv_usec/100000);
        END(outp);
        break;
    case 'E':
        psecs(ms / 100, outp);
        END(outp);
        break;
    case 'P':
        sprintf(outp,"%d%%", (int) (t*100 / ((ms ? ms : 1))));
        END(outp);
        break;
    case 'W':
        i = r1->ru_nswap - r0->ru_nswap;
        sprintf(outp,"%d", i);
        END(outp);
        break;
    case 'X':
        sprintf(outp,"%d", t == 0 ? 0 : (r1->ru_ixrss-r0->ru_ixrss)/t);
        END(outp);
        break;
    case 'D':
        sprintf(outp,"%d", t == 0 ? 0 :
          (r1->ru_idrss+r1->ru_isrss-(r0->ru_idrss+r0->ru_isrss))/t);
        END(outp);
        break;
    case 'K':
```

```
            sprintf(outp,"%d", t == 0 ? 0 :
              ((r1->ru_ixrss+r1->ru_isrss+r1->ru_idrss) -
              (r0->ru_ixrss+r0->ru_idrss+r0->ru_isrss))/t);
            END(outp);
            break;
        case 'M':
            sprintf(outp,"%d", r1->ru_maxrss/2);
            END(outp);
            break;
        case 'F':
            sprintf(outp,"%d", r1->ru_majflt-r0->ru_majflt);
            END(outp);
            break;
        case 'R':
            sprintf(outp,"%d", r1->ru_minflt-r0->ru_minflt);
            END(outp);
            break;
        case 'I':
            sprintf(outp,"%d", r1->ru_inblock-r0->ru_inblock);
            END(outp);
            break;
        case 'O':
            sprintf(outp,"%d", r1->ru_oublock-r0->ru_oublock);
            END(outp);
            break;
        case 'C':
            sprintf(outp,"%d+%d", r1->ru_nvcsw-r0->ru_nvcsw,
                r1->ru_nivcsw-r0->ru_nivcsw );
            END(outp);
            break;
        }
        }
        *outp = '\0';
}
static void
tvadd(tsum, t0, t1)
    struct timeval *tsum, *t0, *t1;
{
    tsum->tv_sec = t0->tv_sec + t1->tv_sec;
    tsum->tv_usec = t0->tv_usec + t1->tv_usec;
    if (tsum->tv_usec > 1000000)
        tsum->tv_sec++, tsum->tv_usec -= 1000000;
}
static void
tvsub(tdiff, t1, t0)
    struct timeval *tdiff, *t1, *t0;
{
    tdiff->tv_sec = t1->tv_sec - t0->tv_sec;
    tdiff->tv_usec = t1->tv_usec - t0->tv_usec;
    if (tdiff->tv_usec < 0)
        tdiff->tv_sec--, tdiff->tv_usec += 1000000;
```

```
}
static void
psecs(l,cp)
long l;
register char *cp;
{
    register int i;
    i = l / 3600;
    if (i) {
    sprintf(cp,"%d:", i);
    END(cp);
    i = l % 3600;
    sprintf(cp,"%d%d", (i/60) / 10, (i/60) % 10);
    END(cp);
    } else {
    i = l;
    sprintf(cp,"%d", i / 60);
    END(cp);
    }
    i %= 60;
    *cp++ = ':';
    sprintf(cp,"%d%d", i / 10, i % 10);
}
#endif
/*
 *      N R E A D
 */
Nread( fd, buf, count )
{
    struct sockaddr_in from;
    int len = sizeof(from);
    register int cnt;
    if( udp ) {
    cnt = recvfrom( fd, (char *) buf, count, 0, (struct sockaddr *) &from, &len );
    } else {
    if( b_flag )
        cnt = mread( fd, buf, count );/* fill buf */
    else
        cnt = read( fd, buf, count );
    }
    return(cnt);
}
/*
 *      N W R I T E
 */
Nwrite( fd, buf, count )
{
    register int cnt;
    if( udp ) {
again:
    cnt = sendto( fd, (char *) buf, count, 0, (struct sockaddr *) &sinhim,
```

```
sizeof(sinhim) );
    if( cnt<0 && errno == ENOBUFS ) {
        delay(18000);
        errno = 0;
        goto again;
    }
    } else {
    cnt = write( fd, buf, count );
    }
    return(cnt);
}
delay(us)
{
    struct timeval tv;
    tv.tv_sec = 0;
    tv.tv_usec = us;
    (void)select( 1, (fd_set *)0, (fd_set *)0, (fd_set *)0, &tv );
    return(1);
}
/*
 *      M R E A D
 *
 * This function performs the function of a read(II) but will
 * call read(II) multiple times in order to get the requested
 * number of characters.  This can be necessary because
 * network connections don't deliver data with the same
 * grouping as it is written with.  Written by Robert S. Miles, BRL.
 */
int
mread(fd, bufp, n)
int fd;
register char*bufp;
unsignedn;
{
    register unsignedcount = 0;
    register intnread;
    do {
    nread = read(fd, bufp, n-count);
    if(nread < 0)  {
        perror("ttcp_mread");
        return(-1);
    }
    if(nread == 0)
        return((int)count);
    count += (unsigned)nread;
    bufp += nread;
     } while(count < n);
    return((int)count);
}
#if sgi
static void
```

```
tvsub(tdiff, t1, t0)
    struct timeval *tdiff, *t1, *t0;
{
    tdiff->tv_sec = t1->tv_sec - t0->tv_sec;
    tdiff->tv_usec = t1->tv_usec - t0->tv_usec;
    if (tdiff->tv_usec < 0)
    tdiff->tv_sec--, tdiff->tv_usec += 1000000;
}
#endif
```

APPENDIX E: NTTCP TEST RESULTS

This Appendix contains the 48 measured data transfers for each of the test runs for the NTTCP testing. The data rates in the following tables are in megabits per second.

Number of source Buffers (bytes)	Memory Buffer 65536 (run 1)	Memory Buffer 8192 (run 1)	Memory Buffer 65536 (run 2)	Memory Buffer 8192 (run 2)	Memory Buffer 65536 (run 3)	Memory Buffer 8192 (run 3)
1024	5.1150	5.1736	5.2250	5.2280	5.2709	5.2514
2048	5.2637	5.2054	5.1764	5.2155	5.2378	5.1939
4096	5.2094	5.2126	5.2209	4.8967	5.1927	5.2041
512	5.1829	5.2977	5.1582	5.2667	5.0722	5.3015

Table 14: Test Run 1: Without Packet Filter PC, NTTCP Run 1-3

Number of source Buffers (bytes)	Memory Buffer 65536 (run 4)	Memory Buffer 8192 (run 4)	Memory Buffer 65536 (run 5)	Memory Buffer 8192 (run 5)	Memory Buffer 65536 (run 6)	Memory Buffer 8192 (run 6)
1024	5.2515	5.3422	5.2276	5.1720	5.2351	5.2351
2048	5.2693	5.1803	5.2325	5.1550	5.2278	5.2278
4096	5.2457	5.2055	5.2522	5.2435	5.1359	5.1359
512	5.2423	5.1020	5.2221	5.2311	5.2846	5.2846

Table 15: Test Run 1: Without Packet Filter PC, NTTCP Run 4-6

Number of source Buffers (bytes)	Memory Buffer 65536 (run 1)	Memory Buffer 8192 (run 1)	Memory Buffer 65536 (run 2)	Memory Buffer 8192 (run 2)	Memory Buffer 65536 (run 3)	Memory Buffer 8192 (run 3)
1024	0.6362	0.6379	0.6365	0.6371	0.6361	0.6369
2048	0.6356	0.6377	0.6358	0.6358	0.6361	0.6373
4096	0.6362	0.6367	0.6361	0.6365	0.6362	0.6366
512	0.8797	0.6374	0.6360	0.6381	0.6361	0.6381

Table 16: Test Run 2: Packet Filter PC in Place (Outbound), NTTCP Run 1-3

Number of source Buffers (bytes)	Memory Buffer 65536 (run 4)	Memory Buffer 8192 (run 4)	Memory Buffer 65536 (run 5)	Memory Buffer 8192 (run 5)	Memory Buffer 65536 (run 6)	Memory Buffer 8192 (run 6)
1024	0.6360	0.6373	0.6359	0.6370	0.6360	0.6378
2048	0.6360	0.6366	0.6359	0.6364	0.6363	0.6367
4096	0.6360	0.6372	0.6359	0.6370	0.6359	0.6369
512	0.6364	0.6351	0.6354	0.6370	0.6353	0.6373

Table 17: Test Run 2: Packet Filter PC in Place (Outbound), NTTCP Run 4-6

Number of source Buffers (bytes)	Memory Buffer 65536 (run 1)	Memory Buffer 8192 (run 1)	Memory Buffer 65536 (run 2)	Memory Buffer 8192 (run 2)	Memory Buffer 65536 (run 3)	Memory Buffer 8192 (run 3)
1024	0.3840	0.5845	0.6062	0.6085	0.6075	0.6045
2048	0.4597	0.6085	0.5867	0.6084	0.6037	0.5551
4096	0.6070	0.6087	0.6086	0.6086	0.6070	0.6092
512	0.4509	0.5999	0.6076	0.6091	0.6073	0.6096

Table 18: Test Run 3: Packet Filter PC in Place (Inbound), NTTCP Run 1-3

Number of source Buffers (bytes)	Memory Buffer 65536 (run 4)	Memory Buffer 8192 (run 4)	Memory Buffer 65536 (run 5)	Memory Buffer 8192 (run 5)	Memory Buffer 65536 (run 6)	Memory Buffer 8192 (run 6)
1024	0.6075	0.6077	0.6072	0.6090	0.6075	0.6085
2048	0.6068	0.6077	0.6072	0.6076	0.6065	0.6088
4096	0.6081	0.6092	0.6077	0.6084	0.6063	0.6069
512	0.6067	0.6070	0.6087	0.6097	0.6068	0.6069

Table 19: Test Run 3: Packet Filter PC in Place (Inbound), NTTCP Run 4-6

LIST OF REFERENCES

[CARL 92] Carl-Mitchell, Smoot, Quarterman, John S., *Building Internet Firewalls*, UnixWorld Magazine, February 1992.

[CHAP 92] Chapman, D. Brent, *Network (In)Security Through IP Packet Filtering,* Proceedings of the Third Usenix UNIX Security Symposium, pages 63-76, Baltimore, MD, September 1992.

[CHES 94] Cheswick, William R., Bellovin, Steven M., *Firewalls and Internet Security, Repelling the Wily Hacker,* Addison-Wesley Publishing Company, 1994.

[DODD 88] DoD Directive 52000.28, *Security Requirements for Automated Information Systems (AISs),* March 21, 1988.

[DODD 85] DoD Standard 52000.28-STD, *Department of Defense Trusted Computer System Evaluation Criteria,* December 1985.

[FARM 93] Farmer, Dan, Venema, Wietse, *Improving the Security of your Site by Breaking into it,* 1993.

[MADR 92] Madron, Thomas W., *Network Security in the '90s, Issues and Solutions for Managers,* John Wiley & Sons, Inc., 1992.

[NCSC 87] National Computer Security Center, *Trusted Network Interpretation of the Trusted Computer System Evaluation Criteria, Version 1,* NCSC-TG-005, July 1987.

[PFLE 89] Pfleeger, Charles P., *Security in Computing,* Prentice Hall, Inc., 1989.

[RICH 92] Rich, Lyford D., *Unix Security: A Penetration Analysis of Navy Computer Systems*, Master's Thesis, Naval Postgraduate School, Monterey, California, December 1992.

[RUSS 91] Russell, Deborah, Gangemi Sr., G.T., *Computer Security Basics,* O'Reilly & Associates, Inc., 1991.

[SAFF 93] Safford, David R., Schales, Douglas Lee, and Hess, David K., *The TAMU Security Package: An Ongoing Response to Internet Intruders in an Academic Environment,* Proceedings of the Fourth Usenix UNIX Security Symposium, pages 91-118, Santa Clara, CA, October 1993.

[STAN 93] Stang, David J., Moon, Sylvia, *Network Security Secrets,* IDG Books Worldwide, Inc., 1993.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center — 2
   Cameron Station
   Alexandria, VA 22304-6145

2. Dudley Knox Library — 2
   Code 052
   Naval Postgraduate School
   Monterey, CA 93943-5002

3. Jody Schivley — 10
   2828-B Sidney Park Place
   Aberdeen Proving Ground, MD 21005

4. LTC Steven A. Ruegnitz — 2
   127 IMA Det R&D
   17 Muirfield Lane
   Bridgewater, NJ 08807-1269

5. Director — 2
   U.S. Army Research Laboratory
   ATTN: AMSRL-CI (CPT Schivley)
   Aberdeen Proving Ground, MD 21005-5067

6. Director — 1
   U.S. Army Research Laboratory
   ATTN: AMSRL-CI (Mr. Mermegan)
   Aberdeen Proving Ground, MD 21005-5067

7. Director — 1
   U.S. Army Research Laboratory
   ATTN: AMSRL-CI (COL Blake)
   Aberdeen Proving Ground, MD 21005-5067

8. Director — 1
   U.S. Army Research Laboratory
   ATTN: AMSRL-CI (Mr. Breaux)
   Aberdeen Proving Ground, MD 21005-5067

9. Chairman, Code 37 CS — 2
   Computer Science Department
   Naval Postgraduate School
   Monterey, CA 93943

10. Professor Timothy Shimeall, Code CS/Sm — 2
    Computer Science Department
    Naval Postgraduate School
    Monterey, CA 93943-5000

11. Professor G.M. Lundy, Code CS/Ln — 2
    Computer Science Department
    Naval Postgraduate School
    Monterey, CA 93943-5000

12. Professor Roger Stemp, Code CS/Sp — 2
    Computer Science Department
    Naval Postgraduate School
    Monterey, CA 93943-5000

13. David Norman, Code 51 — 2
    Computer Center
    Naval Postgraduate School
    Monterey, CA 93943-5000

14. Hiram Cooke, Code 51 — 2
    Computer Center
    Naval Postgraduate School
    Monterey, CA 93943-5000

15. Jeff Franklin, Code 54 — 2
    ADPSO
    Naval Postgraduate School
    Monterey, CA 93943-5000

16. Sun Microsystems
    Attn: Patrick Barrett and Maria Tapia — 2
    1842 N. Shoreline Blvd.
    MS UMTV80-01
    Mountain View, CA 94043-1100

17. CPT Paul R. Logan 1
USMA DOIM BOPD
West Point, NY 10966

18. Northern Arizona University 2
Dr. Robert Feugate
School of Engineering
Box 52000
Flagstaff, AZ 86001

19. Vivian Leber 1
3198 Shoshone Drive
Lake Havasu City, AZ 86406

20. Dr. David Safford / David Hess 2
Texas A&M Univsersity
Computing and Information Services
College Station, TX 77843-3363